

E6290

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-085546

(43)Date of publication of application : 30.03.1999

(51)Int.Cl.

G06F 9/46

G06F 9/46

(21)Application number : 09-248176

(71)Applicant : HITACHI LTD

(22)Date of filing : 12.09.1997

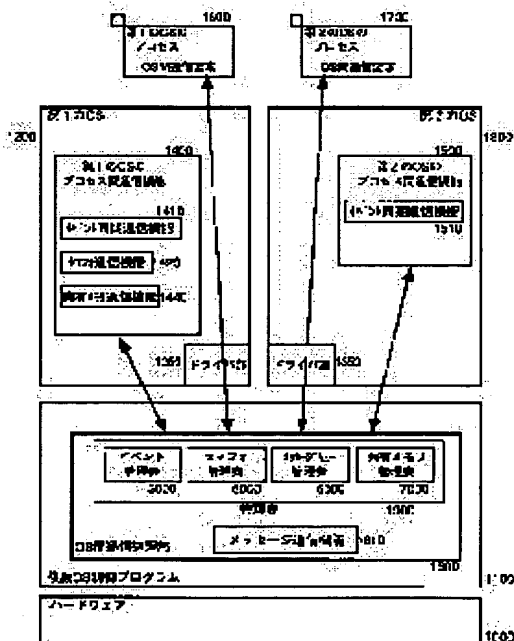
(72)Inventor : INOUE TARO  
 ARAI TOSHIAKI  
 KANEKO SHIGENORI  
 ONO HIROSHI  
 SEKIGUCHI TOMONORI  
 SHIBATA TAKASHI

## (54) INTER-PROCESS COMMUNICATING METHOD ON HETEROGENEOUS OS

## (57)Abstract:

**PROBLEM TO BE SOLVED:** To perform process synchronization and communication between processes on respective OSs by providing a communication means to communicate each other on 1st and 2nd OSs which simultaneously operate on one computer.

**SOLUTION:** A management table 1900 which records the name of an inter-OS communication object and the correspondence of an object handle about an inter-process communication function of a 1st OS 1200 and an object handle about an inter-process communication function of a 2nd OS 1300 is provided in each inter-OS communication object. By the table 1900, a communication request that specifies the names of inter-OS communication objects which are issued from processes of the 1st OS and the 2nd OS can be converted into a request to an inter-process communication function of the OS 1200 or the OS 1300. That is, for instance, an inter-OS communication request that specifies a semaphore can be converted into a semaphore communication request for the OS 1200.



## LEGAL STATUS

[Date of request for examination]

09.03.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision  
of rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2000 Japanese Patent Office

E6290

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-85546

(43)公開日 平成11年(1999) 3月30日

(51)Int.Cl.<sup>8</sup>

G 0 6 F 9/46

識別記号

3 5 0

3 4 0

F I

G 0 6 F 9/46

3 5 0

3 4 0 A

審査請求 未請求 請求項の数45 O L (全 38 頁)

(21)出願番号

特願平9-248176

(22)出願日

平成9年(1997) 9月12日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 井上 太郎

神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作所システム開発研究所内

(72)発明者 新井 利明

神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作所システム開発研究所内

(72)発明者 金子 茂則

茨城県日立市大みか町五丁目2番1号 株式会社日立製作所大みか工場内

(74)代理人 弁理士 小川 勝男

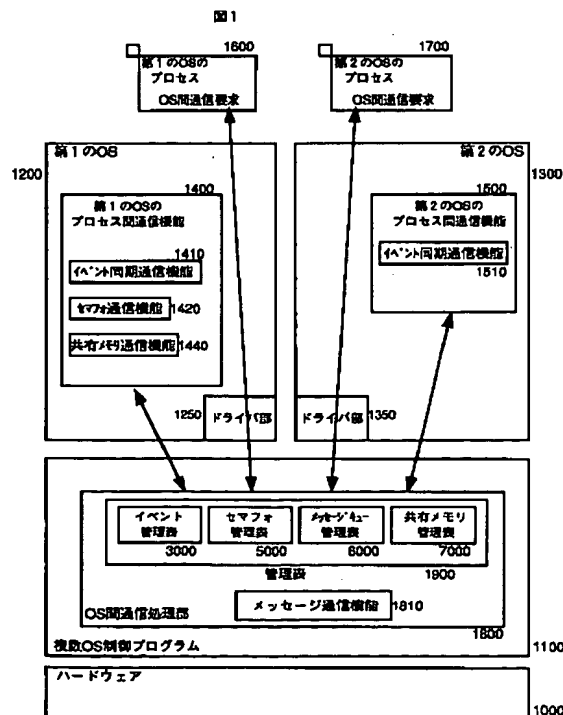
最終頁に続く

(54)【発明の名称】 異種OS上プロセス間通信方法

(57)【要約】

【課題】 1台の計算機上で複数のオペレーティングシステム(OS)が動作するとき、それぞれのOS上で動作するプロセス間でのプロセス間通信(イベント同期、セマフォ、メッセージ、共有メモリ)を可能とする。

【解決手段】 OS間通信オブジェクト(イベント、セマフォ、メッセージキュー、共有メモリ等)毎に、OS間通信オブジェクトの名前と第1のOSのプロセス間通信機能に関するオブジェクトハンドルと第2のOSのプロセス間通信機能に関するオブジェクトハンドルの対応を記録する管理表を設ける。この管理表により、第1のOS上のプロセスおよび第2のOS上のプロセスから発行されたOS間通信オブジェクトの名前を指定した通信要求を、第1のOSあるいは第2のOSのプロセス間通信機能への要求へと変換する。



**【特許請求の範囲】**

【請求項 1】 1 台の計算機上で第 1 のオペレーティングシステム（OS）および第 2 の OS が同時に動作する計算機システムにおいて、

第 1 の OS 上のプロセスおよび第 2 の OS 上のプロセスが相互に通信する手段を有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 2】 請求項 1 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスおよび第 2 の OS 上のプロセスが相互に通信する手段はイベント同期によるプロセス間通信であることを特徴とする異種 OS 上プロセス間通信方法。

【請求項 3】 請求項 2 の異種 OS 上プロセス間通信方法において、

第 1 の OS および第 2 の OS が、それぞれ、イベント同期によるプロセス間通信機能を有する時、  
ウェイトしている第 1 の OS 上のプロセスを第 2 の OS 上のプロセスがポストし、ウェイトしている第 2 の OS 上のプロセスを第 1 の OS 上のプロセスがポストする手段を有することを特徴とする計算機システム異種 OS 上プロセス間通信方法。

【請求項 4】 請求項 3 の異種 OS 上プロセス間通信方法において、

第 1 の OS のイベント同期によるプロセス間通信機能に関するイベントオブジェクトのハンドルと、第 2 の OS のイベント同期によるプロセス間通信機能に関するイベントオブジェクトのハンドルと、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信のためのイベントの名前とのイベント管理表を、第 1 の OS および第 2 の OS を制御するための複数 OS 制御プログラムの中に有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 5】 請求項 4 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのオープン要求を当該イベントの名前を指定して行う場合、

当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがなければ、第 1 の OS のイベント同期によるプロセス間通信機能におけるオープン要求を発行し、その結果得られるイベントオブジェクトのハンドルおよび当該イベントの名前を当該イベント管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 6】 請求項 4 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのウェイトの要求を行う場合、

上記のイベント管理表を用いて第 1 の OS のイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、

上記の獲得したイベントオブジェクトハンドルを用いて、第 1 の OS のイベント同期によるプロセス間通信機能におけるウェイトの要求を発行するステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 7】 請求項 4 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのポストの要求を行う場合、

上記のイベント管理表を用いて第 2 の OS のイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、

第 1 の OS のコンテキストから第 2 の OS のコンテキストに切り替えた後に、上記の獲得したイベントオブジェクトハンドルを用いて、第 2 の OS のイベント同期によるプロセス間通信機能におけるポストの要求を発行するステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 8】 請求項 4 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのクローズの要求を行う場合、

上記のイベント管理表を用いて第 1 の OS のイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、

上記の獲得したイベントオブジェクトハンドルを用いて、第 1 の OS のイベント同期によるプロセス間通信機能におけるクローズの要求を発行するステップと、  
イベント管理表の当該エントリを無効とするステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 9】 請求項 4 の異種 OS 上プロセス間通信方法において、

第 2 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのオープン要求を当該イベントの名前を指定して行う場合、

当該イベント管理表をサーチして、当該イベントの名前

と同一のイベントの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、当該イベント管理表をサーチして、当該イベントの名前と同一のイベントの名前が格納されたエントリがなければ、第2のOSのイベント同期によるプロセス間通信機能におけるオープン要求を発行し、その結果得られるイベントオブジェクトのハンドルおよび当該イベントの名前を当該イベント管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項10】請求項4の異種OS上プロセス間通信方法において、第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのウェイトの要求を行う場合、上記のイベント管理表を用いて第2のOSのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、上記の獲得したイベントオブジェクトハンドルを用いて、第2のOSのイベント同期によるプロセス間通信機能におけるウェイトの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項11】請求項4の異種OS上プロセス間通信方法において、第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのポストの要求を行う場合、上記のイベント管理表を用いて第1のOSのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、上記の獲得したイベントオブジェクトハンドルを用いて、第1のOSのイベント同期によるプロセス間通信機能におけるポストの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項12】請求項4の異種OS上プロセス間通信方法において、第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のイベント同期によるプロセス間通信におけるイベントのクローズの要求を行う場合、上記のイベント管理表を用いて第2のOSのイベント同期によるプロセス間通信のイベントオブジェクトハンドルを得るステップと、上記の獲得したイベントオブジェクトハンドルを用いて、第2のOSのイベント同期によるプロセス間通信機能におけるクローズの要求を発行するステップと、

イベント管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項13】請求項1の異種OS上プロセス間通信方法において、第1のOS上のプロセスおよび第2のOS上のプロセスが相互に通信する手段はセマフォであることを特徴とする異種OS上プロセス間通信方法。

【請求項14】請求項13の異種OS上プロセス間通信方法において、第1のOSがセマフォ機能を有し、第2のOSがセマフォ機能を有さない時、第1のOS上のプロセスが発行する、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォ関連要求を、第1のOSが有するセマフォ機能を用いて処理するステップと、第2のOS上のプロセスが発行する、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォ関連要求を、第1のOSが有するセマフォ機能を用いて処理するステップとを有することを特徴とする異種OS上プロセス間通信方法。

【請求項15】請求項14の異種OS上プロセス間通信方法において、第1のOSのセマフォ機能に関するセマフォオブジェクトのハンドルと、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォの名前とのセマフォ管理表を、第1のOSおよび第2のOSを制御するための複数OS制御プログラムの中に有することを特徴とする異種OS上プロセス間通信方法。

【請求項16】請求項15の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのオープンの要求を当該セマフォの名前を指定して行う場合、

当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがなければ、第1のOSのセマフォ機能におけるオープン要求を発行し、その結果得られるセマフォオブジェクトのハンドルおよび当該セマフォの名前を当該セマフォ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項17】請求項15の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における

セマフォのウェイトの要求を行う場合、  
上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるセマフォのウェイトの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項18】請求項15の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのカウントアップの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるセマフォのカウントアップの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項19】請求項15の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのクローズの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるクローズの要求を発行するステップと、

セマフォ管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項20】請求項15の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのオープン要求を当該セマフォの名前を指定して行う場合、

当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該セマフォ管理表をサーチして、当該セマフォの名前と同一のセマフォの名前が格納されたエントリがなければ、第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、第1のOSのセマフォ機能におけるオープン要求を発行し、その結果得られるセマフォオブジェクトのハンドルおよび当該セマフォの名前を当該セマフォ管理表の新たなエントリに格納し、そのエ

ントリの記述子を返すステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項21】請求項15の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのウェイトの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキストに切り替え後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるセマフォのウェイトの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項22】請求項15の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのカウントアップの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるセマフォのカウントアップの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項23】請求項15の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるセマフォのクローズの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSのセマフォによるプロセス間通信のセマフォオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、上記の獲得したセマフォオブジェクトハンドルを用いて、第1のOSのセマフォ機能におけるクローズの要求を発行するステップと、

セマフォ管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項24】請求項1の異種OS上プロセス間通信方法において、

第1のOS上のプロセスおよび第2のOS上のプロセスが相互に通信する手段はメッセージキューを用いたメッセージであることを特徴とする異種OS上プロセス間通信方法。

【請求項 2 5】請求項 2 4 の異種 OS 上プロセス間通信方法において、

第 1 の OS がメッセージ機能を有さず、第 2 の OS もメッセージ機能を有さない時、

第 1 の OS 上のプロセスが発行する、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージ関連要求を処理するステップと、  
第 2 の OS 上のプロセスが発行する、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージ関連要求を処理するステップとを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 2 6】請求項 2 5 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージキューの名称を管理するメッセージキュー管理表を、第 1 の OS および第 2 の OS を制御するための複数 OS 制御プログラムの中に有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 2 7】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージキューのオープン要求を当該メッセージキューの名称を指定して行う場合、

当該メッセージキュー管理表をサーチして、当該メッセージキューの名称と同一のメッセージキューの名称が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該メッセージキュー管理表をサーチして、当該メッセージキューの名称と同一のメッセージキューの名称が格納されたエントリがなければ、新たなメッセージキューを確保して、当該メッセージキューの名称を当該メッセージキュー管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 2 8】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージ送信の要求を行う場合、

第 1 の OS 上のプロセスの空間上のメッセージデータを、メッセージキューが存在する空間上のバッファへコピーするステップと、

メッセージヘッダを割り当てて、それをメッセージキューのメッセージヘッダリストの最後に連結するステップと、

当該メッセージキューに対するメッセージ到着をウェイトしている第 2 の OS 上のプロセスをポストするステップを有することを特徴とする異種 OS 上プロセス間通信

方法。

【請求項 2 9】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージ受信の要求を行う場合、

当該受信要求において指定したメッセージキューの最初のメッセージを、当該第 1 の OS 上のプロセスの空間の当該受信要求において指定されたアドレスへコピーするステップと、

当該受信要求において指定したメッセージキューにメッセージがなかった場合には、当該メッセージキューに対するメッセージ到着をウェイトするステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 3 0】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 1 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージキューのクローズの要求を行う場合、

メッセージキュー管理表の当該エントリを無効とするステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 3 1】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 2 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージキューのオープン要求を当該メッセージキューの名称を指定して行う場合、

当該メッセージキュー管理表をサーチして、当該メッセージキューの名称と同一のメッセージキューの名称が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該メッセージキュー管理表をサーチして、当該メッセージキューの名称と同一のメッセージキューの名称が格納されたエントリがなければ、新たなメッセージキューを確保して、当該メッセージキューの名称を当該メッセージキュー管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種 OS 上プロセス間通信方法。

【請求項 3 2】請求項 2 6 の異種 OS 上プロセス間通信方法において、

第 2 の OS 上のプロセスが、第 1 の OS 上のプロセスと第 2 の OS 上のプロセスの間のプロセス間通信におけるメッセージ送信の要求を行う場合、

第 2 の OS 上のプロセスの空間上のメッセージデータを、メッセージキューが存在する空間上のバッファへコピーするステップと、

メッセージヘッダを割り当てて、それをメッセージキューのメッセージヘッダリストの最後に連結するステップと、

当該メッセージキューに対するメッセージ到着をウェイトしている第1のOS上のプロセスをポストするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項33】請求項26の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるメッセージ受信の要求を行う場合、

当該受信要求において指定したメッセージキューの最初のメッセージを、当該第2のOS上のプロセスの空間の当該受信要求において指定されたアドレスへコピーするステップと、

当該受信要求において指定したメッセージキューにメッセージがなかった場合には、当該メッセージキューに対するメッセージ到着をウェイトするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項34】請求項26の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信におけるメッセージキューのクローズの要求を行う場合、

メッセージキュー管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項35】請求項1の異種OS上プロセス間通信方法において、

第1のOS上のプロセスおよび第2のOS上のプロセスが相互に通信する手段は共有メモリであることを特徴とする異種OS上プロセス間通信方法。

【請求項36】請求項35の異種OS上プロセス間通信方法において、

第1のOSが共有メモリ機能を有し、第2のOSが共有メモリ機能を有さない時、

第1のOS上のプロセスが発行する、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリ関連要求を、第1のOSが有する共有メモリ機能を用いて処理するステップと、

第2のOS上のプロセスが発行する、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリ関連要求を、第1のOSが有する共有メモリ機能を用いて処理するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項37】請求項36の異種OS上プロセス間通信方法において、

第1のOSの共有メモリに関する共有メモリオブジェクトのハンドルと、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリの名前との共有メモリ管理表を、第1のOSおよび第2のOSを制御するための複数OS制御プログラムの中に

有することを特徴とする異種OS上プロセス間通信方法。

【請求項38】請求項37の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリのオープン要求を当該共有メモリの名前を指定して行う場合、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがなければ、第1のOSの共有メモリ機能におけるオープン要求を発行し、その結果得られる共有メモリオブジェクトのハンドルおよび当該共有メモリの名前を当該共有メモリ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項39】請求項37の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピングの要求を行う場合、

上記のセマフォ管理表を用いて第1のOSの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能における共有メモリの仮想空間マッピングの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項40】請求項37の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピング解除の要求を行う場合、

上記の共有メモリ管理表を用いて第1のOSの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能における共有メモリの仮想空間マッピング解除の要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項41】請求項37の異種OS上プロセス間通信方法において、

第1のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリのクローズの要求を行う場合、

上記の共有メモリ管理表を用いて第1のOSの共有メモ



りによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、  
上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能におけるクローズの要求を発行するステップと、  
共有メモリ管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項42】請求項37の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリのオープン要求を当該共有メモリの名前を指定して行う場合、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがあれば、そのエントリの記述子を返すステップと、

当該共有メモリ管理表をサーチして、当該共有メモリの名前と同一の共有メモリの名前が格納されたエントリがなければ、第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、第1のOSの共有メモリ機能におけるオープン要求を発行し、その結果得られる共有メモリオブジェクトのハンドルおよび当該共有メモリの名前を当該共有メモリ管理表の新たなエントリに格納し、そのエントリの記述子を返すステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項43】請求項37の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピングの要求を行う場合、  
上記の共有メモリ管理表を用いて第1のOSの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキストに切り替え後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能における共有メモリの仮想空間マッピングの要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項44】請求項37の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリの仮想空間マッピング解除の要求を行う場合、

上記の共有メモリ管理表を用いて第1のOSの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキス

トに切り替えた後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能における共有メモリの仮想空間マッピング解除の要求を発行するステップを有することを特徴とする異種OS上プロセス間通信方法。

【請求項45】請求項37の異種OS上プロセス間通信方法において、

第2のOS上のプロセスが、第1のOS上のプロセスと第2のOS上のプロセスの間のプロセス間通信における共有メモリのクローズの要求を行う場合、

上記の共有メモリ管理表を用いて第1のOSの共有メモリによるプロセス間通信の共有メモリオブジェクトハンドルを得るステップと、

第2のOSのコンテキストから第1のOSのコンテキストに切り替えた後に、上記の獲得した共有メモリオブジェクトハンドルを用いて、第1のOSの共有メモリ機能におけるクローズの要求を発行するステップと、

共有メモリ管理表の当該エントリを無効とするステップを有することを特徴とする異種OS上プロセス間通信方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、1台の計算機上で同時に動作する異なるオペレーティングシステム上のプロセスが相互に同期および通信する方法に関する。

【0002】

【従来の技術】一般的に、計算機では1つのオペレーティングシステム(OS)が動作し、それが計算機の各種資源(CPU、メモリ、2次記憶装置等)を管理している。このOSには様々な種類のものがあり、それぞれに特徴がある。即ち、オンライン処理に向けたOS、リアルタイム処理向けのOS、グラフィカルユーザインタフェース(GUI)に優れたOS等である。すると1台の計算機上に複数のOSを動作させようとする要求が生まれる。このための技術としては、仮想計算機あるいはマイクロカーネルがある。

【0003】仮想計算機は、主に汎用計算機の世界で実現されている技術である。仮想計算機制御プログラムがすべての計算機資源を管理し、それを各仮想計算機に対して分割専有したり、仮想化して与えたりするのである。

【0004】一方、マイクロカーネルでは、マイクロカーネルの上にOS機能を実現するOSサーバを構築し、ユーザはそのサーバを通して計算機を利用する。このOSサーバを各種用意すれば様々なOS環境を利用できる。

【0005】また、計算機のOSが提供する機能として、プロセス間の同期機能および通信機能がある(以下では、「プロセス間の同期機能および通信機能」のことを「プロセス間同期・通信機能」と呼ぶことがある)。

プロセス間の同期機能は、主に複数のプロセスが資源を共有する場合にシリアライズして競合を防止するための機能であり、プロセス間通信機能は、複数のプロセスが互いにデータのやりとりをするための機能である。例えば、プロセス間の同期機能としてはイベント同期機能やセマフォがあり、プロセス間通信機能としてはメッセージや共有メモリがある（「UNIXカーネルの設計」Ma urice J. Bach 著、坂本文・多田好克・村井純 訳、p 304～322、原題「The Design of the UNIX Operating System」）。

#### 【0006】

【発明が解決しようとする課題】以上のような仮想計算機あるいはマイクロカーネル等の技術により1台の計算機上に複数のOSを動作させることが可能となる。すると、例えば、1台の計算機上でリアルタイム処理向けOSとGUIに優れたOSが動作しているとき、リアルタイム処理向けOS上のプロセスが取得したデータをGUIに優れたOS上のプロセスで処理して画面に表示する、といった要求が生まれる。これを実現するためには、同時に動作する複数のOS上のプロセス同士の同期機能および通信機能が必要となる。

【0007】そこで、本発明の第1の目的は、1台の計算機上で複数のOSが動作する場合に、それぞれのOS上のプロセス間でのプロセス間同期・通信機能を実現するための方法を提供することにある。

【0008】また、異種OS上のプロセス間での同期機能あるいは通信機能を実現する場合、このOSが備えているプロセス間同期・通信機能を利用することができれば、実現が容易となる。

【0009】そこで、本発明の第2の目的は、1台の計算機上で複数のOSが動作し、それぞれのOS上のプロセス相互間のプロセス間同期・通信機能を実現する際に、これらのOSがもともと備えているプロセス間同期・通信機能を用いて実現するための方法を提供することにある。

#### 【0010】

【課題を解決するための手段】本発明では、OS間通信オブジェクト（イベント、セマフォ、メッセージキュー、共有メモリ等）毎に、OS間通信オブジェクトの名前と第1のOSのプロセス間通信機能に関するオブジェクトハンドルと第2のOSのプロセス間通信機能に関するオブジェクトハンドルの対応を記録する管理表を設ける。この管理表により、第1のOS上のプロセスおよび第2のOS上のプロセスから発行されたOS間通信オブジェクトの名前を指定した通信要求を、第1のOSあるいは第2のOSのプロセス間通信機能への要求へと変換できる。即ち、例えば、セマフォを指定したOS間通信要求を第1のOSのセマフォ通信要求に変換できるのである。これにより、例えば、イベント同期によるプロセス間通信の場合では、ウェイトしている第1のOS上の

プロセスを第2のOS上のプロセスがポストし、ウェイトしている第2のOS上のプロセスを第1のOS上のプロセスがポストすることが可能となる。

#### 【0011】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。

【0012】図1は、本発明の実施例を実施する計算機システムにおけるOS間通信の概要を示す図である。ハードウェア1000は、この計算機システムを構成するハードウェアであり、CPU、メモリ、入出力装置等から成る。複数OS制御プログラム1100は、この計算機上で複数のOS（この例では第1のOS1200および第2のOS1300）が動作するための各種の制御をする部分で、ハードウェア1000の初期化および分割専有処理、CPUのスケジューリング、割り込み処理等を行う。OS間通信処理部1800は、複数OS制御プログラム1100の中に存在し、管理表1900を用いてこの計算機システム上で動作するOSのOS間通信の処理を行う。管理表1900は、OS間通信オブジェクト対応に存在し、イベント管理表3000、セマフォ管理表5000、メッセージキュー管理表6000、共有メモリ管理表7000がある。そして、OS間通信処理部1800には、メッセージ通信機能1810がある。このメッセージ通信機能1810は、メッセージによるOS間通信の処理を行う。第1のOS1200および第2のOS1300は、この計算機システム上で動作するオペレーティングシステム（OS）で、例えば、第1のOSがグラフィカルユーザインタフェース（GUI）に優れたOSであり、第2のOSがリアルタイム性に優れたOSであるという具合である。これらのOSは各々ドライバ部1250およびドライバ部1350を含んでおり、これらドライバを経由してOS間通信処理部1800とアクセスする。そして、それぞれのOSには、第1のOSのプロセス間通信機能1400および第2のOSのプロセス間通信機能1500がある。これらは、それぞれのOS自身がその上で動作するプロセスに対するプロセス間通信の処理を行う部分であり、第1のOSのプロセス間通信機能1400は、イベント同期通信機能1410、セマフォ通信機能1420、共有メモリ通信機能1440から成る。第2のOSのプロセス間通信機能1500は、イベント同期通信機能1510から成る。第1のOSのプロセス1600は第1のOS1200の上で動作するプロセスであり、第2のOSのプロセス1700は第2のOS1300の上で動作するプロセスである。第1のOSのプロセス1600あるいは第2のOSのプロセス1700からOS間通信オブジェクトのオープンが要求がドライバ部1250あるいはドライバ部1350を経由して出されると、OS間通信処理部1800は、この要求で指定されたOS間通信オブジェクトの名前を管理表1900から探し、これと同じ名前のO

S間通信オブジェクトのエントリがあればその記述子を返し、同じ名前のOS間通信オブジェクトがなければ、第1のOSのプロセス間通信機能1400に対してオープン要求を行う。そして、返されたオブジェクトハンドルとこのOS間通信オブジェクトの名前を管理表1900に登録し、そのエントリの記述子を返す。

【0013】図2には、第1のOSのプロセス1600および第2のOSのプロセス1700のプロセスのアドレス空間の構成を示す。第1のOSのプロセス1600のアドレス空間2100および第2のOSのプロセス1700のプロセスのアドレス空間2200は、下半分が各アプリケーション領域2300である。そして、上半分はシステム領域2400であり、第1のOSのプロセス1600および第2のOSのプロセス1700の両者から共通な領域である。このシステム領域2400には、複数OS制御部1100およびOS間通信処理部1800が存在する。

【0014】以下では、まず、イベント同期に関連した処理について説明する。

【0015】図3は、管理表1900の中のイベント管理表3000を示す図である。イベント管理表3000は、OS間通信処理部1800の初期処理時に確保される。イベント管理表3000は、エントリ有効フラグ3010、イベントの名前3020、第1のOSのイベントオブジェクトハンドル3030、第1のOSのイベントオブジェクトハンドル有効フラグ3040、第2のOSのイベントオブジェクトハンドル3050、第2のOSのイベントオブジェクトハンドル有効フラグ3060から成る。この場合は、第1のOS1200および第2のOS1300がともにイベント同期通信機能を有するので、第1のOS1200のイベント同期通信機能のオブジェクトハンドルを格納するのがイベントオブジェクトハンドル3030であり、イベントオブジェクトハンドル有効フラグ3040はそれの有効か無効かを示すフラグである。そして、第2のOS1300のイベント同期通信機能のオブジェクトハンドルを格納するのが第2のOSのイベントオブジェクトハンドル3050で、第2のOSのイベントオブジェクトハンドル有効フラグ3060はそれの有効か無効かを示すフラグである。また、エントリ有効フラグ3010は、このエントリが有効か無効かを示すフラグである。イベントの名前3020は、OS間通信オブジェクトであるところのイベントの名前を格納する。

【0016】OSの中には、プロセス間のイベント同期のためにECB（イベントコントロールブロック）を使用するものもあるが、そのような場合には、第1のOSのイベントオブジェクトハンドル3030あるいは第2のOSのイベントオブジェクトハンドル3050に、ECBのアドレスを格納する。

【0017】図4は、第1のOSのプロセス1600が

発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果としてイベント記述子を得る。以降のイベント同期通信関連のシステムコールでは、このイベント記述子を用いることになる。まず、イベント管理表3000を調べて、エントリ有効フラグ3010がONで、かつ、指定されたイベントの名前とイベントの名前3020が同じであるエントリを探す（ステップ4010）。イベント管理表3000にこのようなエントリがあれば、既にこの名前前のイベントはオープンされていることになる。そこで、このエントリについて、第1のOSのイベントオブジェクトハンドル有効フラグ3040を調べ（ステップ4090）、ONなら、そのエントリのイベント記述子を返して終了する（ステップ4100）。OFFなら、第1のOS1200のイベント同期通信機能1410のオープンのシステムコールをこのイベントの名前で発行することにより、第1のOS1200のイベント同期通信機能1410におけるオブジェクトハンドルを獲得し（ステップ4110）、獲得したオブジェクトハンドルを第1のOSのイベントオブジェクトハンドル3030にセットし（ステップ4120）、第1のOSのイベントオブジェクトハンドル有効フラグ3040をONにし（ステップ4130）、そのエントリのイベント記述子を返して終了する（ステップ4140）。

【0018】一方、ステップ4010で、エントリ有効フラグ3010がONで、かつ、指定されたイベントの名前とイベントの名前3020が同じであるエントリが、イベント管理表3000になければ、エントリ有効フラグ3010がOFFのエントリを探す（ステップ4020）。なければ、イベント管理表3000のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ4030）。エントリ有効フラグ3010がOFFのエントリがあれば、第1のOS1200のイベント同期通信1410のオープンのシステムコールをこのイベントの名前で発行することにより、第1のOS1200のイベント同期通信機能1410のオブジェクトハンドルを獲得し（ステップ4040）、獲得したオブジェクトハンドルを第1のOSのイベントオブジェクトハンドル3030にセットし（ステップ4050）、第1のOSのイベントオブジェクトハンドル有効フラグ3040をONにし（ステップ4060）、パラメータで指定されたイベントの名前をイベントの名前3020に格納し（ステップ4065）、エントリ有効フラグ3010をONにして（ステップ4070）、そのエントリのイベント記述子を返して終了する（ステップ4080）。

【0019】図5は、第1のOSのプロセス1600が発行したイベント同期通信のウェイト要求システムコー

ルの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第1のOSのイベントオブジェクトハンドル3030を用いて、第1のOS1200のイベント同期通信機能1410のウェイト要求のシステムコールを発行し（ステップ4210）、リターンする（ステップ4220）。

【0020】図6は、第1のOSのプロセス1600が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してポスト要求を行う。まず、第2のOS1300へのコンテキスト切替えを行い（ステップ4310）、当該ポスト要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第2のOSのイベントオブジェクトハンドル3050を用いて、第2のOS1300のイベント同期通信機能1510のポスト要求のシステムコールを発行する（ステップ4320）。そして、第1のOS1200へのコンテキスト切替えを行い（ステップ4330）、リターンする（ステップ4340）。

【0021】図7は、第1のOSのプロセス1600が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第1のOSのイベントオブジェクトハンドル3030を用いて、第1のOS1200のイベント同期通信機能1410のクローズ要求のシステムコールを発行し（ステップ4410）、第1のOSのイベントオブジェクトハンドル有効フラグ3040をOFFにする（ステップ4420）。第2のOSのイベントオブジェクトハンドル有効フラグ3060がOFFなら、エントリ有効フラグ3010をOFFにして当該エントリを無効にして（ステップ4440）、リターンする（ステップ4450）。一方、第2のOSのイベントオブジェクトハンドル有効フラグ3060がONなら、リターンする（ステップ4450）。

【0022】図8は、第2のOSのプロセス1700が発行したイベント同期通信のオープン要求システムコールの処理のフローである。このシステムコールは、イベントの名前をパラメータに指定してオープン要求を行い、その結果としてイベント記述子を得る。以降のイベント同期通信関連のシステムコールでは、このイベント記述子を用いることになる。まず、イベント管理表3000を調べて、エントリ有効フラグ3010がONで、かつ、指定されたイベントの名前とイベントの名前30

20が同じであるエントリを探す（ステップ4510）。イベント管理表3000にこのようなエントリがあれば、既にこの名前のイベントはオープンされていることになる。そこで、このエントリについて、第2のOSのイベントオブジェクトハンドル有効フラグ3060を調べ（ステップ4590）、ONなら、そのエントリのイベント記述子を返して終了する（ステップ4600）。OFFなら、第2のOS1300のイベント同期通信1510のオープンのシステムコールをこのイベントの名前で発行することにより、第2のOS1300のイベント同期通信機能1510のオブジェクトハンドルを獲得し（ステップ4610）、獲得したオブジェクトハンドルを第2のOSのイベントオブジェクトハンドル3050にセットし（ステップ4620）、第2のOSのイベントオブジェクトハンドル有効フラグ3060をONにし（ステップ4630）、そのエントリのイベント記述子を返して終了する（ステップ4640）。

【0023】一方、ステップ4010で、エントリ有効フラグ3010がONで、かつ、指定されたイベントの名前とイベントの名前3020が同じであるエントリが、イベント管理表3000になければ、エントリ有効フラグ3010がOFFのエントリを探す（ステップ4520）。なければ、イベント管理表3000のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ4530）。エントリ有効フラグ3010がOFFのエントリがあれば、第1のOS1200のイベント同期通信機能1510のオープンのシステムコールをこのイベントの名前で発行することにより、第2のOS1300のイベント同期通信機能1510のオブジェクトハンドルを獲得し（ステップ4540）、獲得したオブジェクトハンドルを第2のOSのイベントオブジェクトハンドル3050にセットし（ステップ4550）、第2のOSのイベントオブジェクトハンドル有効フラグ3060をONにし（ステップ4560）、パラメータで指定されたイベントの名前をイベントの名前3020に格納し（ステップ4565）、エントリ有効フラグ3010をONにして（ステップ4570）、そのエントリのイベント記述子を返して終了する（ステップ4580）。

【0024】図9は、第2のOSのプロセス1700が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してウェイト要求を行う。まず、当該ウェイト要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第2のOSのイベントオブジェクトハンドル3050を用いて、第2のOS1300のイベント同期通信機能1510のウェイト要求のシステムコールを発行し（ステップ4710）、リターンする（ステップ4720）。

【0025】図10は、第2のOSのプロセス1700が発行したイベント同期通信のポスト要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してポスト要求を行う。まず、第1のOS1200へのコンテキスト切替えを行い（ステップ4810）、当該ポスト要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第1のOSのイベントオブジェクトハンドル3030を用いて、第1のOS1200のイベント同期通信機能1410のポスト要求のシステムコールを発行する（ステップ4820）。そして、第2のOS1300へのコンテキスト切替えを行い（ステップ4830）、リターンする（ステップ4840）。

【0026】図11は、第2のOSのプロセス1700が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。このシステムコールは、イベント記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたイベント記述子に対応するエントリをイベント管理表3000からみつけ、そのエントリにおける第2のOSのイベントオブジェクトハンドル3050を用いて、第2のOS1300のイベント同期通信機能1510のクローズ要求のシステムコールを発行し（ステップ4910）、第2のOSのイベントオブジェクトハンドル有効フラグ3060をOFFにする（ステップ4920）。第1のOSのイベントオブジェクトハンドル有効フラグ3040がOFFなら、エントリ有効フラグ3010をOFFにして当該エントリを無効にして（ステップ4940）、リターンする（ステップ4950）。一方、第2のOSのイベントオブジェクトハンドル有効フラグ3040がONなら、リターンする（ステップ4950）。

【0027】以上の図3から図11に示した処理により、第1のOS1200および第2のOS1300が、それぞれ、イベント同期によるプロセス間通信機能を有する時、ウェイトしている第1のOS上のプロセス1600を第2のOS上のプロセス1700がポストし、ウェイトしている第2のOS上のプロセス1700を第1のOS上のプロセス1600がポストすることが可能となる。

【0028】次に、セマフォに関連した処理について説明する。

【0029】図12は、管理表1900の中のセマフォ管理表5000を示す図である。セマフォ管理表5000は、OS間通信処理部1800の初期処理時に確保される。セマフォ管理表5000は、エントリ有効フラグ5010、セマフォの名前5020、第1のOSのセマフォオブジェクトハンドル5030、第1のOSのセマフォオブジェクトハンドル有効フラグ5040から成る。この場合は、第1のOS1200がセマフォ通信機

能を有するので、第1のOS1200のセマフォ通信機能のオブジェクトハンドルを格納するのがセマフォオブジェクトハンドル5030であり、セマフォオブジェクトハンドル有効フラグ5040はそれの有効か無効かを示すフラグである。また、エントリ有効フラグ5010は、このエントリが有効か無効かを示すフラグである。セマフォの名前5020は、OS間通信オブジェクトであるところのセマフォの名前を格納する。

【0030】図13は、第1のOSのプロセス1600が発行したセマフォ通信のオープン要求システムコールの処理のフローである。このシステムコールは、セマフォの名前をパラメータに指定してオープン要求を行い、その結果としてセマフォ記述子を得る。以降のセマフォ通信関連のシステムコールでは、このセマフォ記述子を用いることになる。第1のOS1200はセマフォ通信機能を有する。まず、セマフォ管理表5000を調べて、エントリ有効フラグ5010がONで、かつ、指定されたセマフォの名前とセマフォの名前5020が同じであるエントリを探す（ステップ5110）。セマフォ管理表5000にこのようなエントリがあれば、既にこの名前のセマフォはオープンされていることになる。そこで、そのエントリのセマフォ記述子を返して終了する（ステップ5190）。

【0031】一方、ステップ5110で、エントリ有効フラグ5010がONで、かつ、指定されたセマフォの名前とセマフォの名前5020が同じであるエントリが、セマフォ管理表5000になければ、エントリ有効フラグ5010がOFFのエントリを探す（ステップ5120）。なければ、セマフォ管理表5000のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ5130）。エントリ有効フラグ5010がOFFのエントリがあれば、第1のOS1200のセマフォ通信のオープンのシステムコールをこのセマフォの名前で発行することにより、第1のOS1200のセマフォ通信機能1420のオブジェクトハンドルを獲得し（ステップ5140）、獲得したオブジェクトハンドルを第1のOSのセマフォオブジェクトハンドル5030にセットし（ステップ5150）、第1のOSのセマフォオブジェクトハンドル有効フラグ5040をONにし（ステップ5160）、パラメータで指定されたセマフォの名前をセマフォの名前5020に格納し（ステップ5165）、エントリ有効フラグ5010をONにして（ステップ5170）、そのエントリのセマフォ記述子を返して終了する（ステップ5180）。

【0032】図14は、第1のOSのプロセス1600が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォウェイト要求を行う。まず、当該セマフォウェイト要求において指定されたセマフォ記述子に対応するエントリを

セマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のセマフォウェイト要求のシステムコールを発行し（ステップ5210）、リターンする（ステップ5220）。

【0033】図15は、第1のOSのプロセス1600が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォカウントアップ要求を行う。まず、当該セマフォカウントアップ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のセマフォカウントアップ要求のシステムコールを発行し（ステップ5310）、リターンする（ステップ5320）。

【0034】図16は、第1のOSのプロセス1600が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のクローズ要求のシステムコールを発行し（ステップ5410）、第1のOSのセマフォオブジェクトハンドル有効フラグ5040をOFFにし（ステップ5420）、エントリ有効フラグ5010をOFFにして当該エントリを無効にして（ステップ5430）、リターンする（ステップ5440）。

【0035】図17は、第2のOSのプロセス1700が発行したセマフォ通信のオープン要求システムコールの処理のフローである。このシステムコールは、セマフォの名前をパラメータに指定してオープン要求を行い、その結果としてセマフォ記述子を得る。以降のセマフォ通信関連のシステムコールでは、このセマフォ記述子を用いることになる。第2のOS1300はセマフォ通信機能を有しない。まず、セマフォ管理表5000を調べて、エントリ有効フラグ5010がONで、かつ、指定されたセマフォの名前とセマフォの名前5020が同じであるエントリを探す（ステップ5510）。セマフォ管理表5000にこのようなエントリがあれば、既にこの名前のセマフォはオープンされていることになる。そこで、そのエントリのセマフォ記述子を返して終了する（ステップ5590）。

【0036】一方、ステップ5510で、エントリ有効フラグ5010がONで、かつ、指定されたセマフォの

名前とセマフォの名前5020が同じであるエントリが、セマフォ管理表5000になければ、エントリ有効フラグ5010がOFFのエントリを探す（ステップ5520）。なければ、セマフォ管理表5000のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ5530）。エントリ有効フラグ5010がOFFのエントリがあれば、第1のOS1200へコンテキスト切替えを行い（ステップ5535）、第1のOS1200のセマフォ通信機能1420のオープンのシステムコールをこのセマフォの名前で発行することにより、第1のOS1200のセマフォ通信機能1420のオブジェクトハンドルを獲得し（ステップ5540）、獲得したオブジェクトハンドルを第1のOSのセマフォオブジェクトハンドル5030にセットし（ステップ5550）、第1のOSのセマフォオブジェクトハンドル有効フラグ5040をONにし（ステップ5560）、パラメータとして指定されたセマフォの名前をセマフォの名前5020に格納し（ステップ5565）、エントリ有効フラグ5010をONにして（ステップ5570）、第2のOS1300へコンテキスト切替えを行い（ステップ5575）、そのエントリのセマフォ記述子を返して終了する（ステップ5580）。

【0037】図18は、第2のOSのプロセス1700が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォウェイト要求を行う。まず、第1のOS1200へのコンテキスト切替えを行い（ステップ5610）、当該セマフォウェイト要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のセマフォウェイト要求のシステムコールを発行し（ステップ5620）、第2のOS1300へのコンテキスト切替えを行い（ステップ5630）、リターンする（ステップ5640）。

【0038】図19は、第2のOSのプロセス1700が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してセマフォカウントアップ要求を行う。まず、第1のOS1200へのコンテキスト切替えを行い（ステップ5710）、当該セマフォカウントアップ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のセマフォカウントアップ要求のシステムコールを発行し（ステップ5720）、第2のOS1300へのコンテキスト切替えを行い（ステップ5730）、リターンする

(ステップ5740)。

【0039】図20は、第2のOSのプロセス1700が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、セマフォ記述子をパラメータに指定してクローズ要求を行う。まず、第1のOS1200へのコンテキスト切替えを行い(ステップ5810)、当該クローズ要求において指定されたセマフォ記述子に対応するエントリをセマフォ管理表5000からみつけ、そのエントリにおける第1のOSのセマフォオブジェクトハンドル5030を用いて、第1のOS1200のセマフォ通信機能1420のクローズ要求のシステムコールを発行し(ステップ5820)、第1のOSのセマフォオブジェクトハンドル有効フラグ5040をOFFにし(ステップ5830)、エントリ有効フラグ5010をOFFにして当該エントリを無効にして(ステップ5840)、第2のOS1300へのコンテキスト切替えを行い(ステップ5850)、リターンする(ステップ5860)。

【0040】以上の図12から図20に示した処理により、第1のOS1200がセマフォ機能を有し、第2のOS1300がセマフォ機能を有さない時、第1のOS上のプロセス1600が発行する、第1のOS上のプロセス1600と第2のOS上のプロセス1700の間のプロセス間通信におけるセマフォ関連要求を、第1のOS1200が有するセマフォ機能を用いて処理することが可能となり、そして、第2のOS上のプロセス1700が発行する、第1のOS上のプロセス1600と第2のOS上のプロセス1700の間のプロセス間通信におけるセマフォ関連要求を、第1のOS1200が有するセマフォ機能を用いて処理することが可能となる。

【0041】次に、メッセージ通信に関連した処理について説明する。

【0042】図21は、メッセージ通信関連のデータ構造を示す図である。メッセージ通信関連のデータ構造はOS間通信処理部1800の中のメッセージ通信機能1810の中にあり、メッセージキュー6000、メッセージヘッダ6010、データバッファ6020から成る。メッセージキュー6000には、メッセージヘッダ6010へのポインタ、当該メッセージキューに対するメッセージを待つプロセスへのポインタおよび管理情報があり、メッセージヘッダのリストが連結される。この管理情報には、連結されたリスト上のメッセージ数と総バイト数等がある。メッセージキュー6000は、OS間通信処理部1800の初期処理時に確保され、その数はメッセージキュー管理表6100のエントリ数と同じになる。メッセージヘッダ6010には、次のメッセージヘッダ6010へのポインタ、メッセージデータへのポインタおよび管理情報が格納されている。この管理情報には、メッセージデータのバイト数等がある。データバッファ6020には、メッセージデータが格納され

る。即ち、メッセージの送り手のメッセージデータがデータバッファ6020にコピーされ、更に、メッセージの受け手にコピーされる。

【0043】図22は、管理表1900の中のメッセージキュー管理表6100を示す図である。メッセージ管理表6100は、OS間通信処理部1800の初期処理時に確保され、そのエントリ数はメッセージキュー6000の数と同じであり、各エントリはメッセージキューに対応する。メッセージ管理表6100は、メッセージキュー使用中フラグ6110、メッセージキューの名前6120、メッセージキューポインタ6130から成る。メッセージキュー使用中フラグ6110は、このエントリに対応するメッセージキューの使用中表示するフラグである。メッセージキューの名前6120は、OS間通信オブジェクトであるところのメッセージキューの名前が格納される。メッセージキューポインタ6130は、このエントリに対応するメッセージキュー6000へのポインタである。このメッセージキューポインタ6130は、メッセージ管理表6100およびメッセージキュー6000の確保時に設定される。

【0044】図23は、第1のOSのプロセス1600が発行したメッセージ通信のオープン要求システムコールの処理のフローである。このシステムコールは、メッセージキューの名前をパラメータに指定してオープン要求を行い、その結果としてメッセージキュー記述子を得る。以降のメッセージ通信関連のシステムコールでは、このメッセージキュー記述子を用いることになる。まず、メッセージキュー管理表6100を調べて、メッセージキュー使用中フラグ6110がONで、かつ、指定されたメッセージキューの名前とメッセージキューの名前6120が同じであるエントリを探す(ステップ6210)。メッセージキュー管理表6100にこのようなエントリがあれば、既にこの名前前のメッセージキューはオープンされていることになる。そこで、そのエントリのメッセージキュー記述子を返して終了する(ステップ6270)。

【0045】一方、ステップ6210で、メッセージキュー使用中フラグ6110がONで、かつ、指定されたメッセージキューの名前とメッセージキューの名前6120が同じであるエントリが、メッセージキュー管理表6100になければ、メッセージキュー使用中フラグ6110がOFFのエントリを探す(ステップ6220)。なければ、メッセージキュー管理表6100のすべてのエントリが使用中、即ち、すべてのメッセージキューが使用中であることになるのでエラーリターンする(ステップ6230)。メッセージキュー使用中フラグ6110がOFFのエントリがあれば、そのエントリのメッセージキュー使用中フラグ6110をONにして(ステップ6240)、パラメータで指定されたメッセージキューの名前をメッセージキューの名前6120に

格納し（ステップ6250）、そのエントリのメッセージキュー記述子を返して終了する（ステップ6260）。

【0046】図24は、第1のOSのプロセス1600が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージが格納される領域のアドレス、送信するメッセージのバイト数をパラメータに指定してメッセージ送信要求を行う。まず、送信メッセージのバイト数が上限を超えるか否かをチェックし（ステップ6310）、超えればエラーリターンする（ステップ6390）。超えなければ、メッセージヘッダ6010およびデータバッファ6020の両者の割り当てが可能か否かをチェックし（ステップ6320）、可能でなければエラーリターンする（ステップ6380）。両者とも割り当て可能であれば、割り当てたデータバッファ6020に送信メッセージをコピーし（ステップ6330）、割り当てたメッセージヘッダ6010をパラメータで指定されたメッセージキュー6000のリストの最後に連結し、当該メッセージキュー6000の管理情報を更新する（ステップ6340）。次に、図4に示したイベント同期通信のオープン要求システムコールおよび図6に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起し（ステップ6360）、リターンする（ステップ6370）。

【0047】図25は、第1のOSのプロセス1600が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージ格納領域アドレスをパラメータに指定してメッセージ受信要求を行う。まず、パラメータで指定されたメッセージキュー6000にメッセージがあるか否かをチェックする（ステップ6410）。あれば、ステップ6430へ行く。なければ、図4に示したイベント同期通信のオープン要求システムコールおよび図5に示したイベント同期通信のウェイト要求システムコールを発行し、当該メッセージキューにメッセージが届くのを待つ（ステップ6420）。そして、メッセージキュー6000に連結されたリストの最初のメッセージヘッダ6010からポイントされるメッセージバッファ6020のメッセージを、パラメータで指定されたメッセージ格納領域アドレスにコピーし（ステップ6430）、当該メッセージヘッダ6010を当該メッセージキュー6000のリストから外し、当該メッセージキュー6000の管理情報を更新し（ステップ6440）、リターンする（ステップ6450）。

【0048】図26は、第1のOSのプロセス1600が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、メッ

セージキュー記述子をパラメータに指定してクローズ要求を行う。パラメータで指定されたメッセージキュー6000にメッセージがあるか否かをチェックする（ステップ6510）。なければ、ステップ6530へ行く。あれば、図4に示したイベント同期通信のオープン要求システムコールおよび図6に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起す（ステップ6520）。そして、メッセージキュー使用中フラグ6110をOFFにして（ステップ6530）、リターンする（ステップ6540）。

【0049】図27は、第2のOSのプロセス1700が発行したメッセージ通信のオープン要求システムコールの処理のフローである。このシステムコールは、メッセージキューの名前をパラメータに指定してオープン要求を行い、その結果としてメッセージキュー記述子を得る。以降のメッセージ通信関連のシステムコールでは、このメッセージキュー記述子を用いることになる。まず、メッセージキュー管理表6100を調べて、メッセージキュー使用中フラグ6110がONで、かつ、指定されたメッセージキューの名前とメッセージキューの名前6120が同じであるエントリを探す（ステップ6610）。メッセージキュー管理表6100にこのようなエントリがあれば、既にこの名前のメッセージキューはオープンされていることになる。そこで、そのエントリのメッセージキュー記述子を返して終了する（ステップ6670）。

【0050】一方、ステップ6210で、メッセージキュー使用中フラグ6110がONで、かつ、指定されたメッセージキューの名前とメッセージキューの名前6120が同じであるエントリが、メッセージキュー管理表6100になければ、メッセージキュー使用中フラグ6110がOFFのエントリを探す（ステップ6620）。なければ、メッセージキュー管理表6100のすべてのエントリが使用中、即ち、すべてのメッセージキューが使用中であることになるのでエラーリターンする（ステップ6630）。メッセージキュー使用中フラグ6110がOFFのエントリがあれば、そのエントリのメッセージキュー使用中フラグ6110をONにして（ステップ6640）、パラメータで指定されたメッセージキューの名前をメッセージキューの名前6120に格納し（ステップ6650）、そのエントリのメッセージキュー記述子を返して終了する（ステップ6660）。

【0051】図28は、第2のOSのプロセス1700が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージが格納される領域のアドレス、送信するメッセージのバイト数をパラメータに指定してメッセージ送信要求を行う。まず、送



信メッセージのバイト数が上限を超えるか否かをチェックし（ステップ6710）、超えればエラーリターンする（ステップ6790）。超えなければ、メッセージヘッダ6010およびデータバッファ6020の両者の割り当てが可能か否かをチェックし（ステップ6720）、可能でなければエラーリターンする（ステップ6780）。両者とも割り当て可能であれば、割り当てたデータバッファ6020に送信メッセージをコピーし

（ステップ6730）、割り当てたメッセージヘッダ6010をパラメータで指定されたメッセージキュー6000のリストの最後に連結し、当該メッセージキュー6000の管理情報を更新する（ステップ6740）。次に、図8に示したイベント同期通信のオープン要求システムコールおよび図10に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こし（ステップ6760）、リターンする（ステップ6770）。

【0052】図29は、第2のOSのプロセス1700が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子、メッセージ格納領域アドレスをパラメータに指定してメッセージ受信要求を行う。まず、パラメータで指定されたメッセージキュー6000にメッセージがあるか否かをチェックする（ステップ6810）。あれば、ステップ6830へ行く。なければ、図8に示したイベント同期通信のオープン要求システムコールおよび図9に示したイベント同期通信のウェイト要求システムコールを発行し、当該メッセージキューにメッセージが届くのを待つ（ステップ6820）。そして、メッセージキュー6000に連結されたリストの最初のメッセージヘッダ6010からポイントされるメッセージバッファ6020のメッセージを、パラメータで指定されたメッセージ格納領域アドレスにコピーし（ステップ6830）、当該メッセージヘッダ6010を当該メッセージキュー6000のリストから外し、当該メッセージキュー6000の管理情報を更新し（ステップ6840）、リターンする（ステップ6850）。

【0053】図30は、第2のOSのプロセス1700が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、メッセージキュー記述子をパラメータに指定してクローズ要求を行う。パラメータで指定されたメッセージキュー6000にメッセージがあるか否かをチェックする（ステップ6910）。なければ、ステップ6930へ行く。あれば、図8に示したイベント同期通信のオープン要求システムコールおよび図10に示したイベント同期通信のポスト要求システムコールを発行し、当該メッセージキューへのメッセージ到着を待つプロセスを起こす（ステップ6920）。そして、メッセージキュー使用中フ

ラグ6110をOFFにして（ステップ6930）、リターンする（ステップ6940）。

【0054】以上の図21から図30に示した処理により、第1のOSがメッセージ機能を有さず、第2のOSもメッセージ機能を有さない時、OS間通信処理部1800上に設けたメッセージキュー6000の介して、第1のOS上のプロセス1600と第2のOS上のプロセス1700の間のプロセス間通信が可能となる。

【0055】次に、共有メモリに関連した処理について説明する。

【0056】図31は、管理表1900の中の共有メモリ管理表7000を示す図である。共有メモリ管理表7000は、OS間通信処理部1800の初期処理時に確保される。共有メモリ管理表7000は、エントリ有効フラグ7010、共有メモリの名前7020、第1のOSの共有メモリオブジェクトハンドル7030、第1のOSの共有メモリオブジェクトハンドル有効フラグ7040から成る。この場合は、第1のOS1200が共有メモリ機能を有するので、第1のOS1200の共有メモリ機能のオブジェクトハンドルを格納するのが共有メモリオブジェクトハンドル7030であり、共有メモリオブジェクトハンドル有効フラグ7040はそれの有効か無効を示すフラグである。また、エントリ有効フラグ7010は、このエントリが有効か無効を示すフラグである。共有メモリの名前7020は、OS間通信オブジェクトであるところの共有メモリの名前を格納する。

【0057】図32は、第1のOSのプロセス1600が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。このシステムコールは、共有メモリの名前をパラメータに指定してオープン要求を行い、その結果として共有メモリ記述子を得る。以降の共有メモリ通信関連のシステムコールでは、この共有メモリ記述子を用いることになる。第1のOS1200は共有メモリ通信機能を有する。まず、共有メモリ管理表7000を調べて、エントリ有効フラグ7010がONで、かつ、指定された共有メモリの名前と共有メモリの名前7020が同じであるエントリを探す（ステップ7110）。共有メモリ管理表7000にこのようなエントリがあれば、既にこの名前前のセマフォはオープンされていることになる。そこで、そのエントリの共有メモリ記述子を返して終了する（ステップ7190）。

【0058】一方、ステップ7110で、エントリ有効フラグ7010がONで、かつ、指定された共有メモリの名前と共有メモリの名前7020が同じであるエントリが、セマフォ管理表7000になければ、エントリ有効フラグ7010がOFFのエントリを探す（ステップ7120）。なければ、共有メモリ管理表7000のすべてのエントリが使用中であることになるのでエラーリターンする（ステップ7130）。エントリ有効フラグ

7010がOFFのエントリがあれば、第1のOS1200の共有メモリ通信機能1440のオープンシステムコールをこの共有メモリの名前で発行することにより、第1のOS1200の共有メモリ通信機能1440のオブジェクトハンドルを獲得し(ステップ7140)、獲得したオブジェクトハンドルを第1のOSの共有メモリオブジェクトハンドル7030にセットし(ステップ7150)、第1のOSの共有メモリオブジェクトハンドル有効フラグ7040をONにし(ステップ7160)、パラメータで指定された共有メモリの名前を共有メモリの名前7020に格納し(ステップ7165)、エントリ有効フラグ7010をONにして(ステップ7170)、そのエントリの共有メモリ記述子を返して終了する(ステップ7180)。

【0059】図33は、第1のOSのプロセス1600が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング要求を行い、マッピングされた仮想空間アドレスを得る。まず、当該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表7000からみつけ、そのエントリにおける第1のOSの共有メモリオブジェクトハンドル7030を用いて、第1のOS1200の共有メモリ通信機能1440の仮想空間マッピング要求のシステムコールを発行し(ステップ7210)、マッピングされた仮想空間アドレスをリターンする(ステップ7220)。

【0060】図34は、第1のOSのプロセス1600が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング解除要求を行う。まず、当該仮想空間マッピング解除要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表7000からみつけ、そのエントリにおける第1のOSの共有メモリオブジェクトハンドル7030を用いて、第1のOS1200の共有メモリ通信機能1440の仮想空間マッピング解除要求のシステムコールを発行し(ステップ7310)、リターンする(ステップ7320)。

【0061】図35は、第1のOSのプロセス1600が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定してクローズ要求を行う。まず、当該クローズ要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表7000からみつけ、そのエントリにおける第1のOSの共有メモリオブジェクトハンドル7030を用いて、第1のOS1200の共有メモリ通信機能1440のクローズ要求のシステムコールを発行し(ステップ7410)、第1のOSの共有メモリオブジェクトハンドル有

効フラグ7040をOFFにし(ステップ7420)、エントリ有効フラグ7010をOFFにして当該エントリを無効にして(ステップ7430)、リターンする(ステップ7440)。

【0062】図36は、第2のOSのプロセス1700が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。このシステムコールは、共有メモリの名前をパラメータに指定してオープン要求を行い、その結果として共有メモリ記述子を得る。以降の共有メモリ通信関連のシステムコールでは、この共有メモリ記述子を用いることになる。第2のOS1300は共有メモリ通信機能を有しない。まず、共有メモリ管理表7000を調べて、エントリ有効フラグ7010がONで、かつ、指定された共有メモリの名前と共有メモリの名前7020が同じであるエントリを探す(ステップ7510)。共有メモリ管理表7000にこのようなエントリがあれば、既にこの名前の共有メモリはオープンされていることになる。そこで、そのエントリの共有メモリ記述子を返して終了する(ステップ7590)。

【0063】一方、ステップ7510で、エントリ有効フラグ7010がONで、かつ、指定された共有メモリの名前と共有メモリの名前7020が同じであるエントリが、共有メモリ管理表7000になければ、エントリ有効フラグ7010がOFFのエントリを探す(ステップ7520)。なければ、共有メモリ管理表7000のすべてのエントリが使用中であることになるのでエラーリターンする(ステップ7530)。エントリ有効フラグ7010がOFFのエントリがあれば、第1のOS1200へコンテキスト切替えを行い(ステップ7535)、第1のOS1200の共有メモリ通信機能1440のオープンシステムコールをこの共有メモリの名前で発行することにより、第1のOS1200の共有メモリ通信機能1440のオブジェクトハンドルを獲得し(ステップ7540)、獲得したオブジェクトハンドルを第1のOSの共有メモリオブジェクトハンドル7030にセットし(ステップ7550)、第1のOSの共有メモリオブジェクトハンドル有効フラグ7040をONにし(ステップ7560)、パラメータとして指定された共有メモリの名前を共有メモリの名前7020に格納し(ステップ7565)、エントリ有効フラグ7010をONにして(ステップ7570)、第2のOS1300へコンテキスト切替えを行い(ステップ7575)、そのエントリの共有メモリ記述子を返して終了する(ステップ7580)。

【0064】図37は、第2のOSのプロセス1700が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング要求を行う。まず、第1のOS1200へのコンテキスト切替えを行い(ステップ7610)、当

該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表 7000 からみつけ、そのエントリにおける第 1 の OS の共有メモリオブジェクトハンドル 7030 を用いて、第 1 の OS 1200 の共有メモリ通信機能 1440 の仮想空間マッピング要求のシステムコールを発行し（ステップ 7620）、第 2 の OS 1300 へのコンテキスト切替えを行い（ステップ 7630）、リターンする（ステップ 7640）。

【0065】図 38 は、第 2 の OS のプロセス 1700 が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定して仮想空間マッピング解除要求を行う。まず、第 1 の OS 1200 へのコンテキスト切替えを行い（ステップ 7710）、当該仮想空間マッピング要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表 7000 からみつけ、そのエントリにおける第 1 の OS の共有メモリオブジェクトハンドル 7030 を用いて、第 1 の OS 1200 の共有メモリ通信機能 1440 の仮想空間マッピング解除要求のシステムコールを発行し（ステップ 7720）、第 2 の OS 1300 へのコンテキスト切替えを行い（ステップ 7730）、リターンする（ステップ 7740）。

【0066】図 39 は、第 2 の OS のプロセス 1700 が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。このシステムコールは、共有メモリ記述子をパラメータに指定してクローズ要求を行う。まず、第 1 の OS 1200 へのコンテキスト切替えを行い（ステップ 7810）、当該クローズ要求において指定された共有メモリ記述子に対応するエントリを共有メモリ管理表 7000 からみつけ、そのエントリにおける第 1 の OS の共有メモリオブジェクトハンドル 7030 を用いて、第 1 の OS 1200 の共有メモリ通信機能 1440 のクローズ要求のシステムコールを発行し（ステップ 7820）、第 1 の OS の共有メモリオブジェクトハンドル有効フラグ 7040 を OFF にし（ステップ 7830）、エントリ有効フラグ 7010 を OFF にして当該エントリを無効にして（ステップ 7840）、第 2 の OS 1300 へのコンテキスト切替えを行い（ステップ 7850）、リターンする（ステップ 7860）。

【0067】以上の図 31 から図 39 に示した処理により、第 1 の OS 1200 が共有メモリ機能を有し、第 2 の OS 1300 が共有メモリ機能を有さない時、第 1 の OS 上のプロセス 1600 が発行する、第 1 の OS 上のプロセス 1600 と第 2 の OS 上のプロセス 1700 の間のプロセス間通信における共有メモリ関連要求を、第 1 の OS 1200 が有する共有メモリ機能を用いて処理することが可能となり、そして、第 2 の OS 上のプロセ

ス 1700 が発行する、第 1 の OS 上のプロセス 1600 と第 2 の OS 上のプロセス 1700 の間のプロセス間通信における共有メモリ関連要求を、第 1 の OS 1200 が有する共有メモリ機能を用いて処理することが可能となる。

【0068】

【発明の効果】 1 台の計算機上で複数のオペレーティングシステム（OS）が動作するとき、それぞれの OS 上で動作するプロセス間でのプロセス間通信（イベント同期、セマフォ、メッセージ、共有メモリ）が可能となる。そして、この通信において、それぞれの OS が持つ自身のプロセス間通信機能を利用するので、実現が容易になる。

【図面の簡単な説明】

【図 1】本発明の実施例を実施する計算機システムにおける OS 間通信の概要を示す図である。

【図 2】第 1 の OS のプロセス 1600 および第 2 の OS のプロセス 1700 のプロセスのアドレス空間の構成を示す図である。

【図 3】管理表 1900 の中のイベント管理表 3000 を示す図である。

【図 4】第 1 の OS のプロセス 1600 が発行したイベント同期通信のオープン要求システムコールの処理のフローである。

【図 5】第 1 の OS のプロセス 1600 が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。

【図 6】第 1 の OS のプロセス 1600 が発行したイベント同期通信のポスト要求システムコールの処理のフローである。

【図 7】第 1 の OS のプロセス 1600 が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。

【図 8】第 2 の OS のプロセス 1700 が発行したイベント同期通信のオープン要求システムコールの処理のフローである。

【図 9】第 2 の OS のプロセス 1700 が発行したイベント同期通信のウェイト要求システムコールの処理のフローである。

【図 10】第 2 の OS のプロセス 1700 が発行したイベント同期通信のポスト要求システムコールの処理のフローである。

【図 11】第 2 の OS のプロセス 1700 が発行したイベント同期通信のクローズ要求システムコールの処理のフローである。

【図 12】管理表 1900 の中のセマフォ管理表 5000 を示す図である。

【図 13】第 1 の OS のプロセス 1600 が発行したセマフォ通信のオープン要求システムコールの処理のフローである。

【図14】第1のOSのプロセス1600が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。

【図15】第1のOSのプロセス1600が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。

【図16】第1のOSのプロセス1600が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。

【図17】第2のOSのプロセス1700が発行したセマフォ通信のオープン要求システムコールの処理のフローである。

【図18】第2のOSのプロセス1700が発行したセマフォ通信のセマフォウェイト要求システムコールの処理のフローである。

【図19】第2のOSのプロセス1700が発行したセマフォ通信のセマフォカウントアップ要求システムコールの処理のフローである。

【図20】第2のOSのプロセス1700が発行したセマフォ通信のクローズ要求システムコールの処理のフローである。

【図21】メッセージ通信関連のデータ構造を示す図である。

【図22】管理表1900の中のメッセージキュー管理表6100を示す図である。

【図23】第1のOSのプロセス1600が発行したメッセージ通信のオープン要求システムコールの処理のフローである。

【図24】第1のOSのプロセス1600が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。

【図25】第1のOSのプロセス1600が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。

【図26】第1のOSのプロセス1600が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。

【図27】第2のOSのプロセス1700が発行したメッセージ通信のオープン要求システムコールの処理のフローである。

【図28】第2のOSのプロセス1700が発行したメッセージ通信のメッセージ送信要求システムコールの処理のフローである。

【図29】第2のOSのプロセス1700が発行したメッセージ通信のメッセージ受信要求システムコールの処理のフローである。

【図30】第2のOSのプロセス1700が発行したメッセージ通信のクローズ要求システムコールの処理のフローである。

【図31】管理表1900の中の共有メモリ管理表70

00を示す図である。

【図32】第1のOSのプロセス1600が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。

【図33】第1のOSのプロセス1600が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。

【図34】第1のOSのプロセス1600が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。

【図35】第1のOSのプロセス1600が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。

【図36】第2のOSのプロセス1700が発行した共有メモリ通信のオープン要求システムコールの処理のフローである。

【図37】第2のOSのプロセス1700が発行した共有メモリ通信の仮想空間マッピング要求システムコールの処理のフローである。

【図38】第2のOSのプロセス1700が発行した共有メモリ通信の仮想空間マッピング解除要求システムコールの処理のフローである。

【図39】第2のOSのプロセス1700が発行した共有メモリ通信のクローズ要求システムコールの処理のフローである。

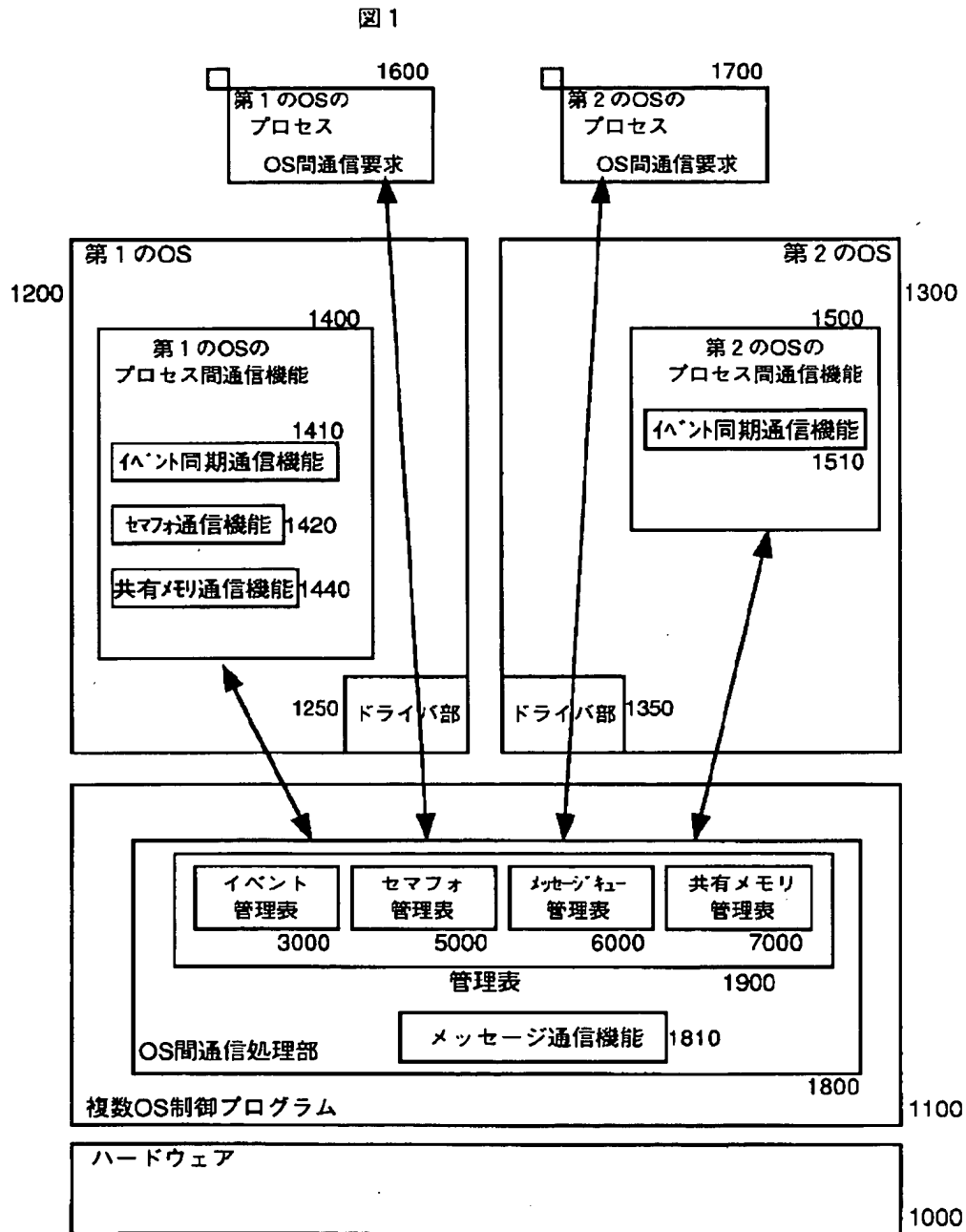
【符号の説明】

1000：ハードウェア、1100：複数OS制御プログラム、1200：第1のOS、1250：ドライバ部、1300：第2のOS、1350：ドライバ部、1400：第1のOSのプロセス間通信機能、1410：イベント同期通信機能、1420：セマフォ通信機能、1440：共有メモリ通信機能、1500：第2のOSのプロセス間通信機能、1510：イベント同期通信機能、1600：第1のOSのプロセス、1700：第2のOSのプロセス、1800：OS間通信処理部、1810：メッセージ通信機能、1900：管理表、2100：第1のOSのプロセス1600のアドレス空間、2200：第2のOSのプロセス1700のアドレス空間、2300：アプリケーション領域、2400：システム領域、3000：イベント管理表、3010：エントリ有効フラグ、3020：イベントの名前、3030：第1のOSのイベントオブジェクトハンドル、3040：第1のOSのイベントオブジェクトハンドル有効フラグ、3050：第2のOSのイベントオブジェクトハンドル、3060：第2のOSのイベントオブジェクトハンドル有効フラグ、5000：セマフォ管理表、5010：エントリ有効フラグ、5020：セマフォの名前、5030：第1のOSのセマフォオブジェクトハンドル、5040：第1のOSのセマフォオブジェクトハンドル有効フラグ、6000：メッセージキュー、60

10:メッセージヘッダ、6020:データバッファ、  
6100:メッセージキュー管理表、6110:メッセ  
ージキュー使用中フラグ6110、6120:メッセ  
ージキューの名前、6130:メッセージキューポイン  
タ、7000:共有メモリ管理表、7010:エントリ

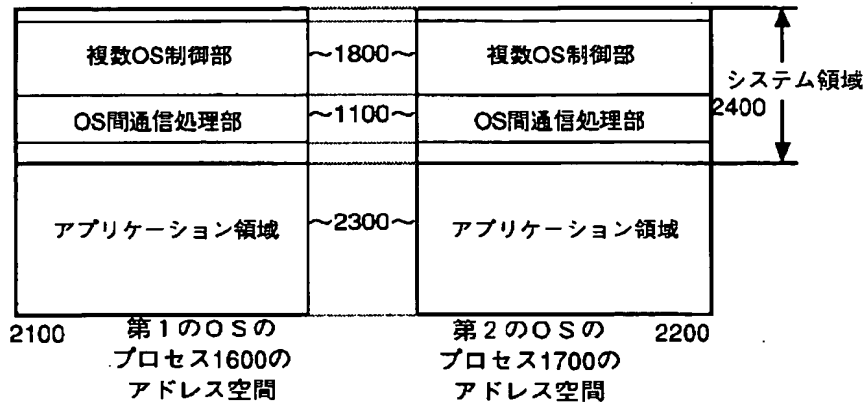
有効フラグ、7020:共有メモリの名前、7030:  
第1のOSの共有メモリオブジェクトハンドル、704  
0:第1のOSの共有メモリオブジェクトハンドル有効  
フラグ。

【図1】



【図2】

図2



【図3】

図3

エントリ有効フラグ	イベントの名前	第1のOSのイベントオブジェクトハンドル	第1のOSのイベントオブジェクトハンドル有効フラグ	第2のOSのイベントオブジェクトハンドル	第2のOSのイベントオブジェクトハンドル有効フラグ

3010      3020      3030      3040      3050      3060

【図12】

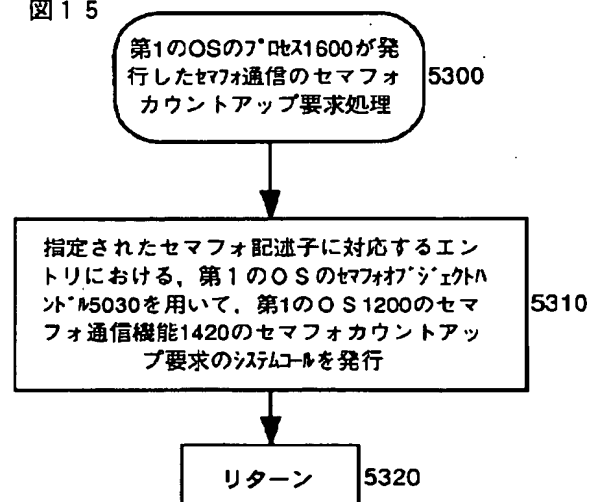
図12

エントリ有効フラグ	セマフォの名前	第1のOSのセマフォオブジェクトハンドル	第1のOSのセマフォオブジェクトハンドル有効フラグ

5010      5020      5030      5040

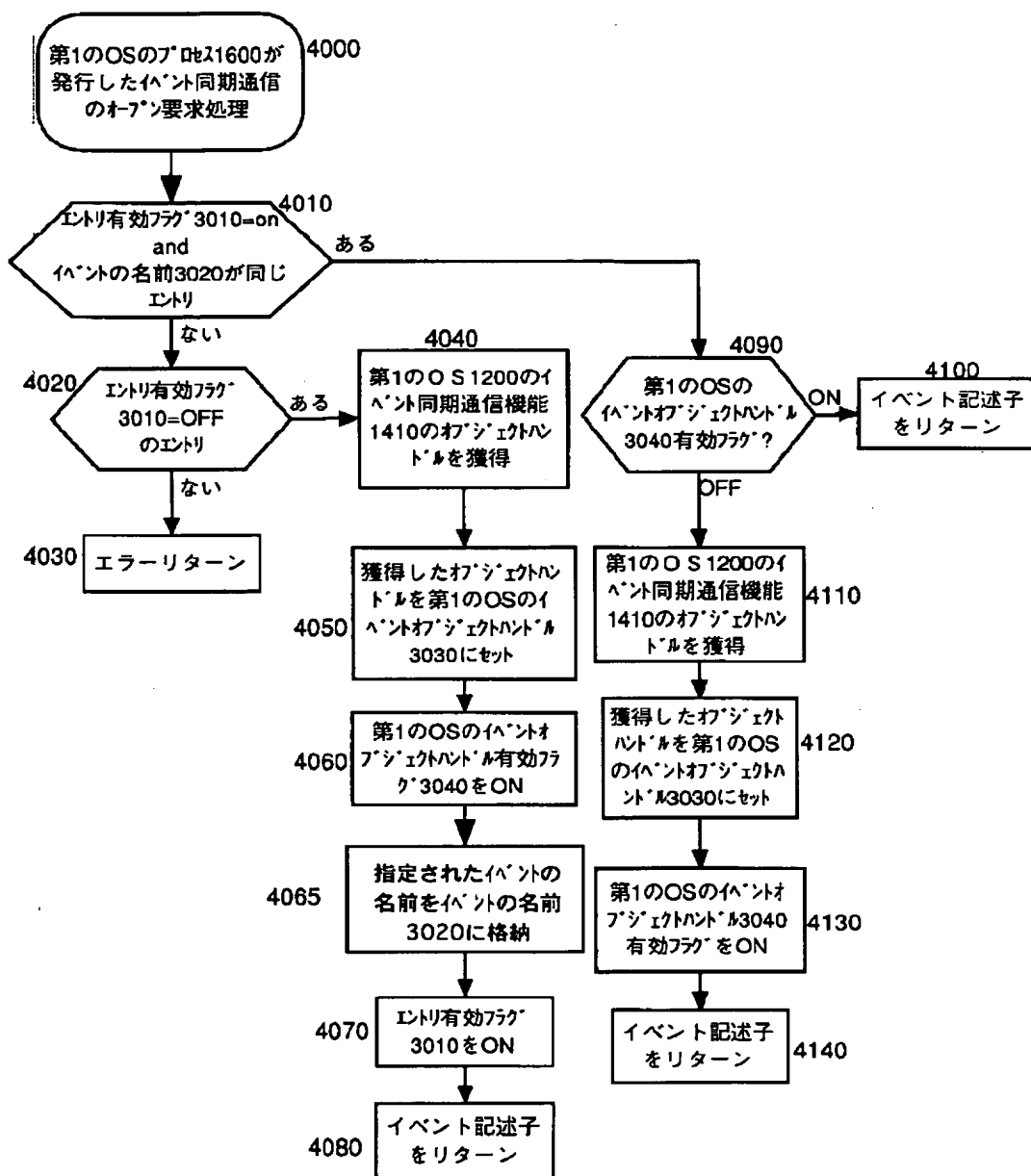
【図15】

図15



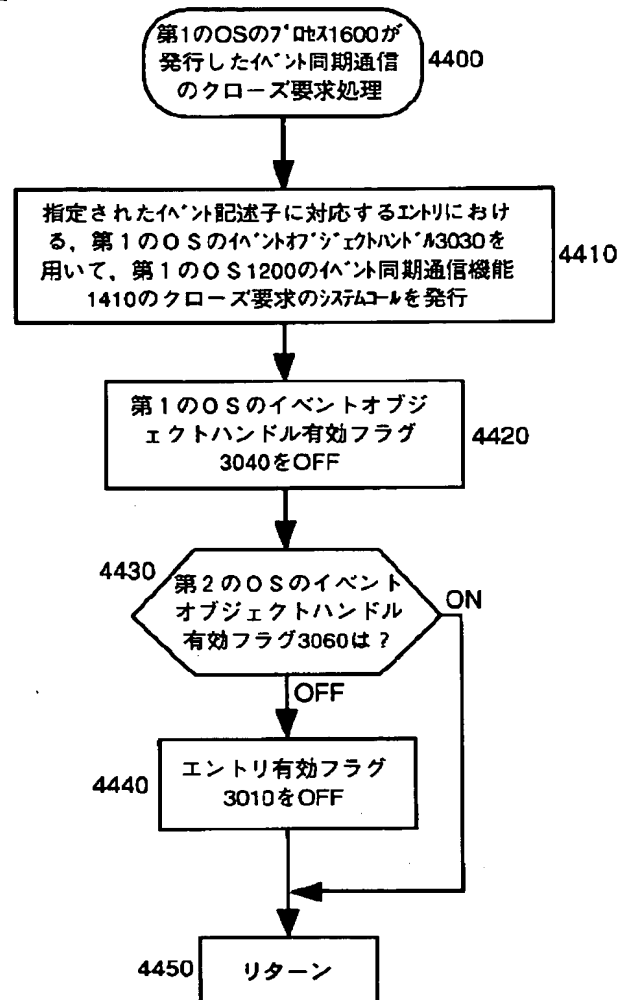
【図4】

図4



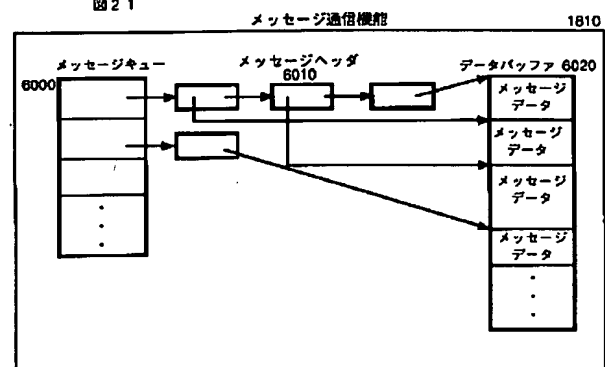
【図 7】

圖 7



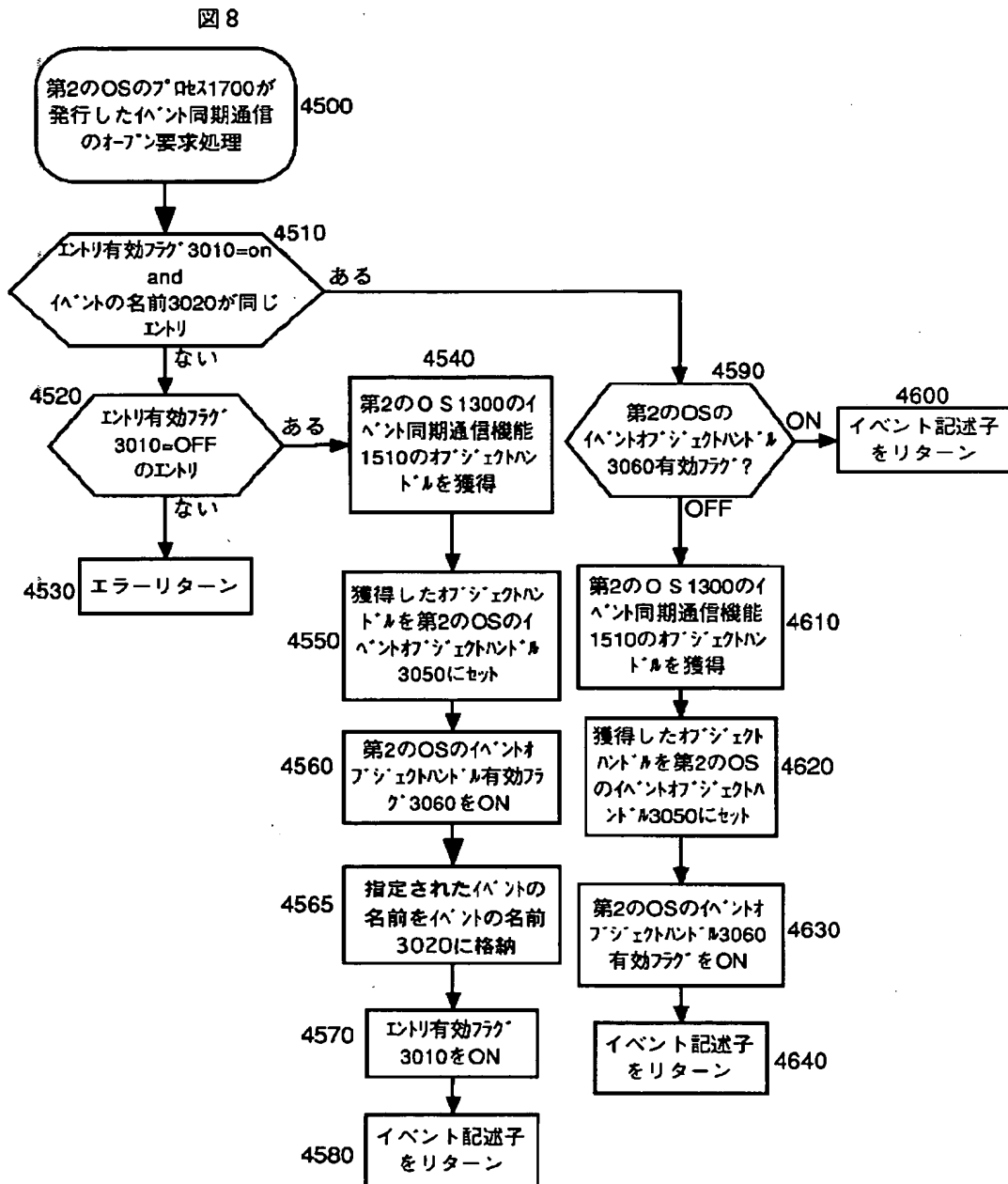
【図 2 1】

21





【図 8】



【図 22】

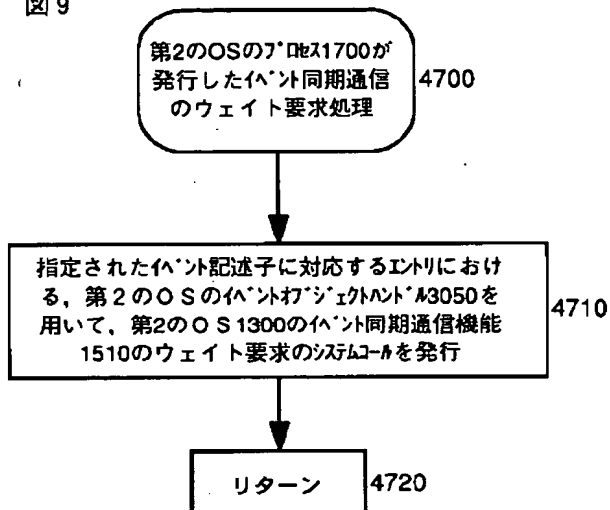
図 22

メッセージキュー 使用中フラグ	メッセージキューの名前	メッセージキューポインタ

6110                  6120                  6130

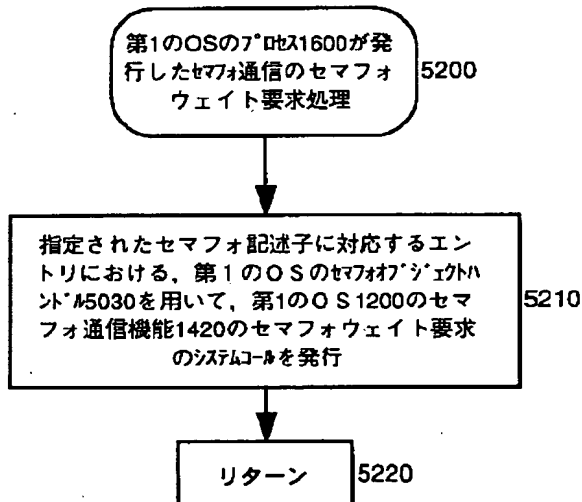
【図 9】

図 9



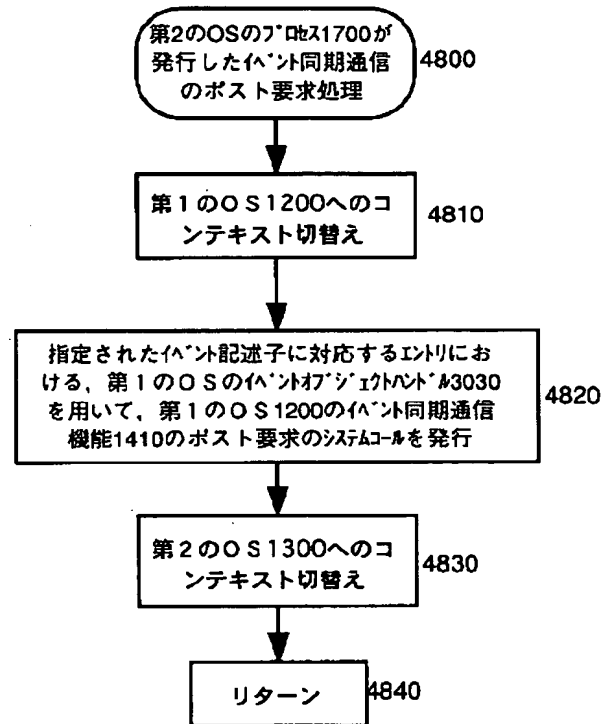
【図 14】

図 14



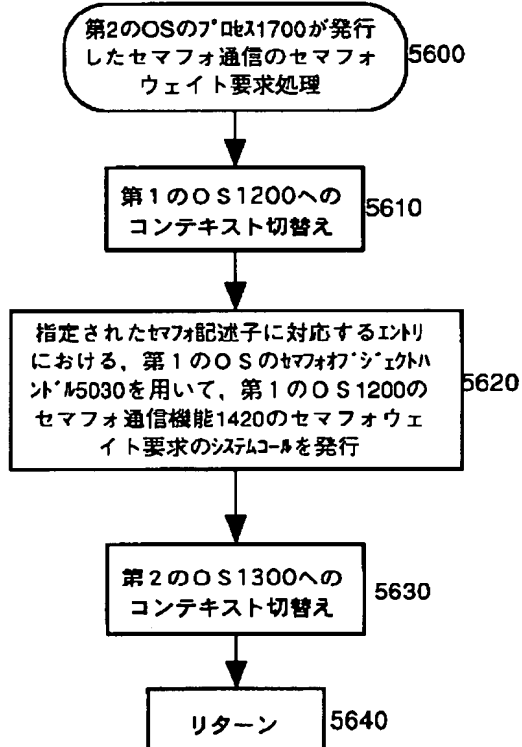
【図 10】

図 10



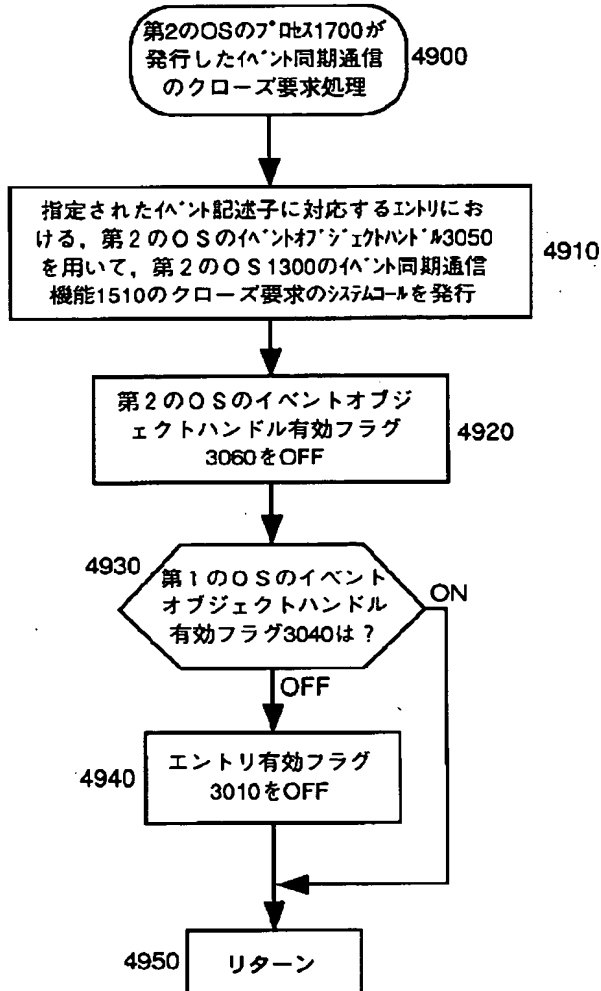
【図 18】

図 18



【図11】

図11



【図31】

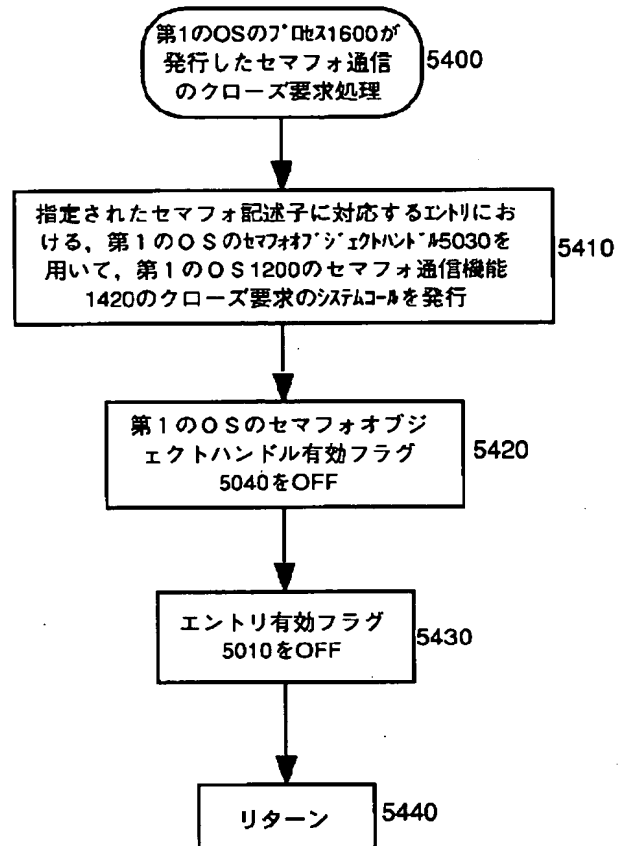
図31

エントリ有効フラグ	共有メモリの名前	第1のOSの共有メモリオブジェクトハンドル	第1のOSの共有メモリオブジェクトハンドル有効フラグ

7010      7020      7030      7040

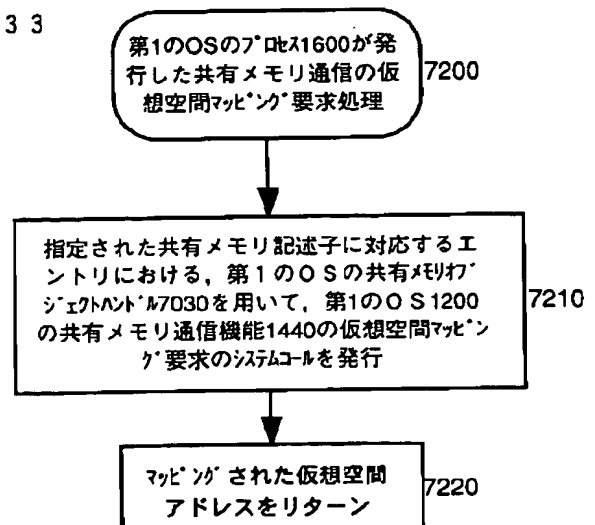
【図16】

図16



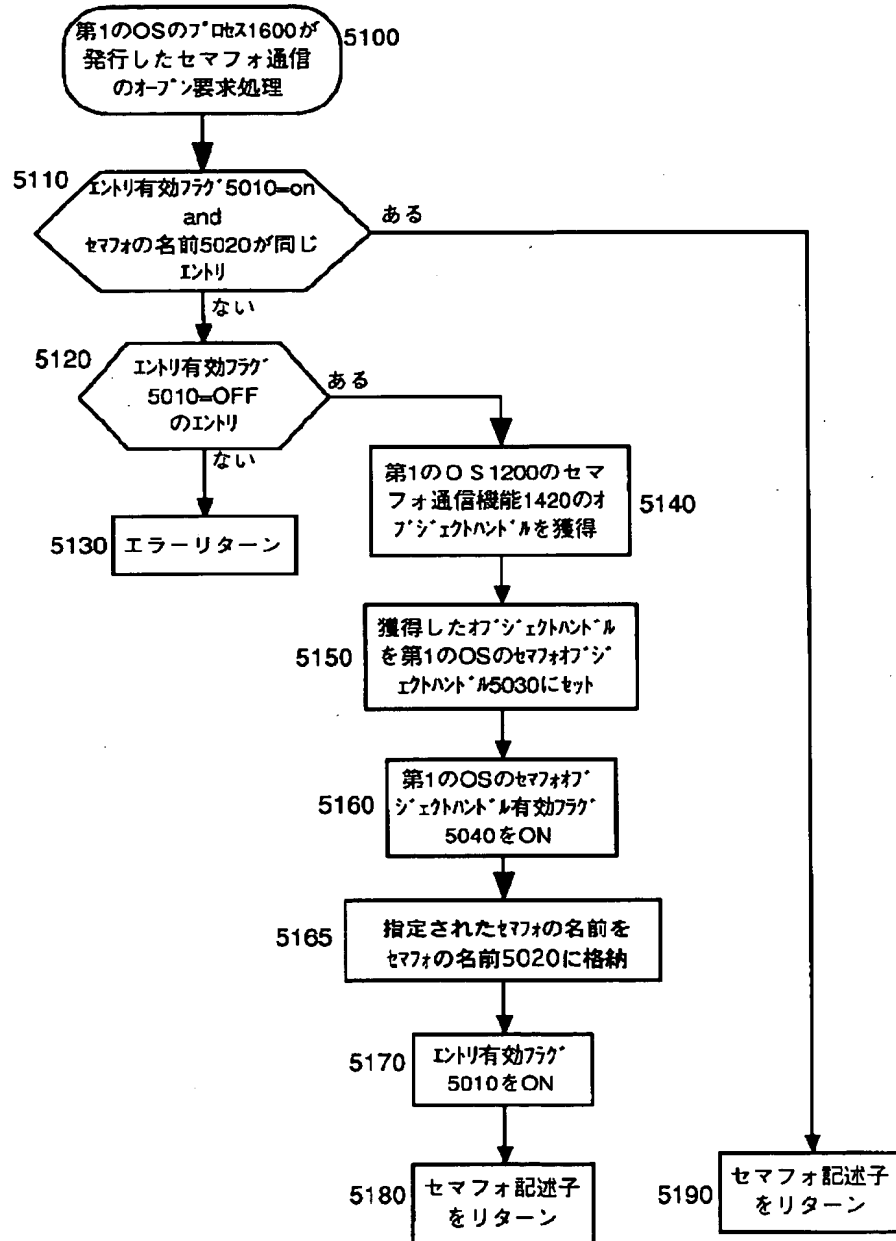
【図33】

図33



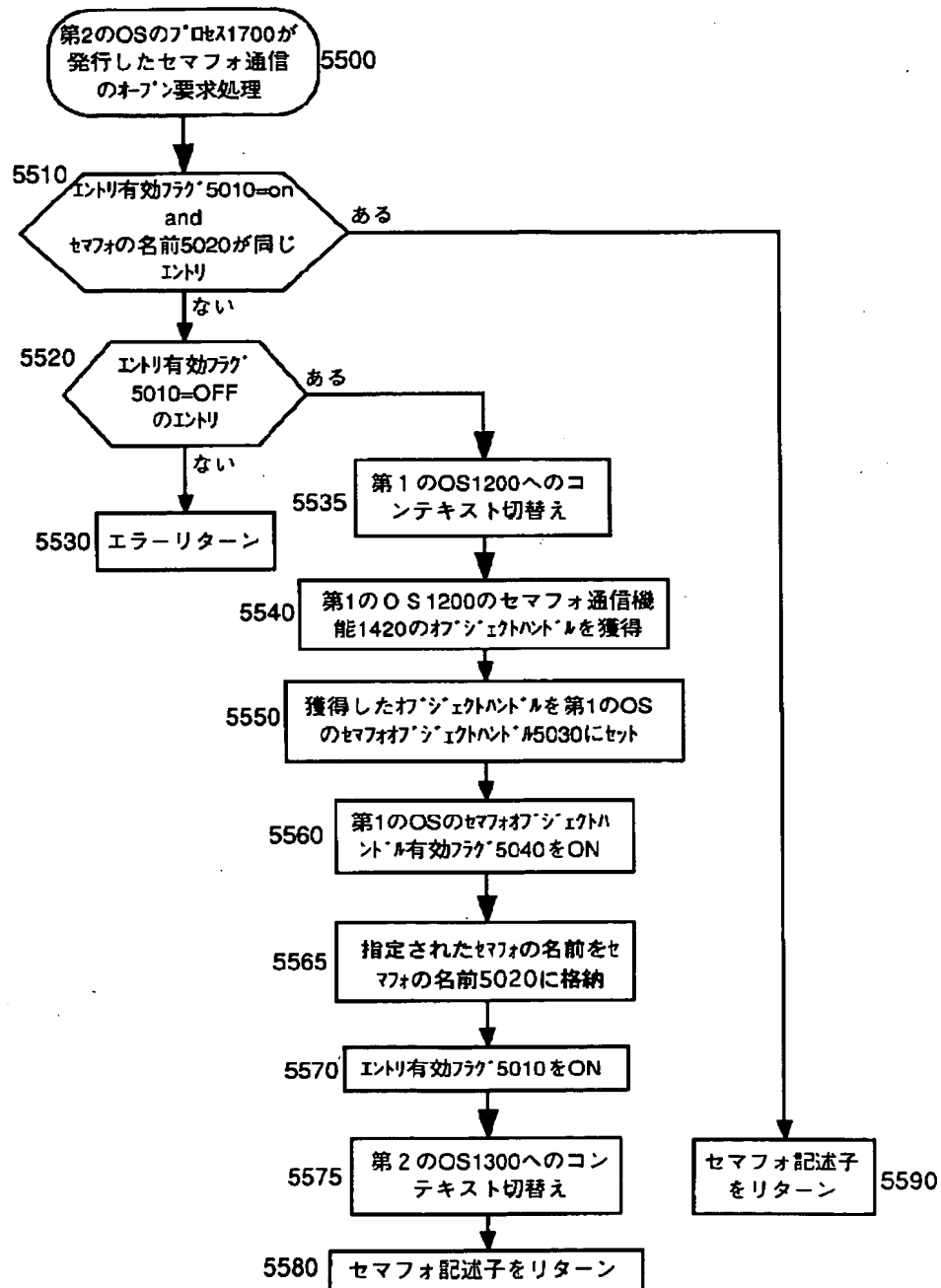
【図 13】

図 13



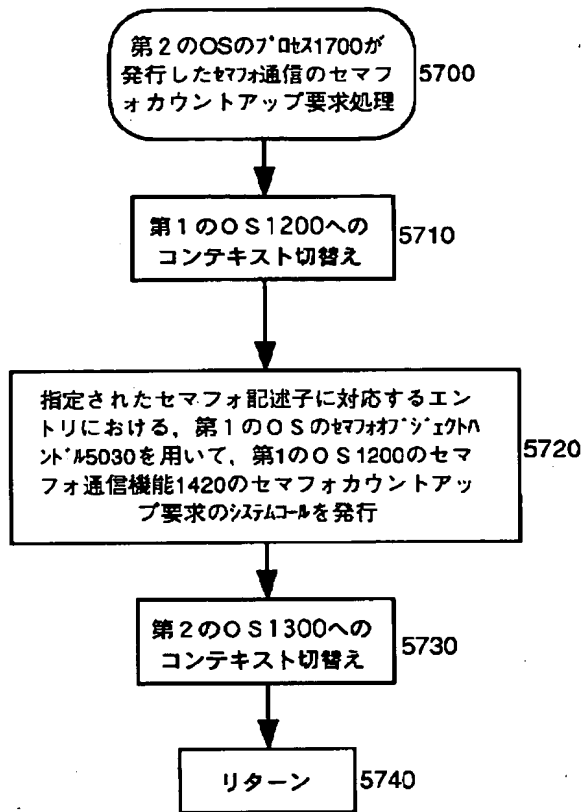
【図17】

図17



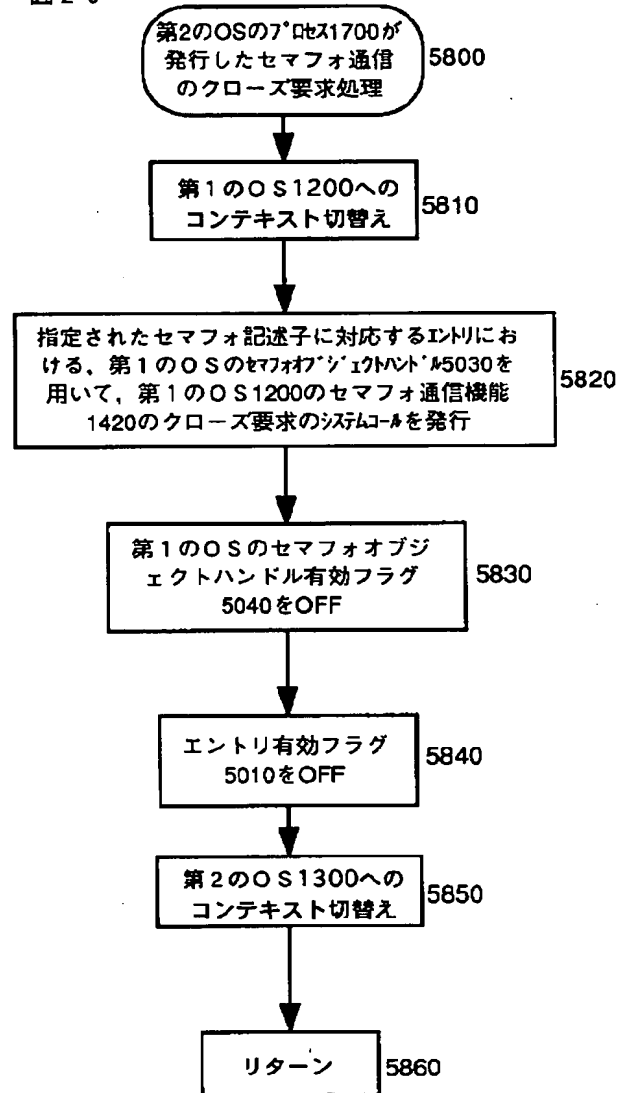
【図19】

図19



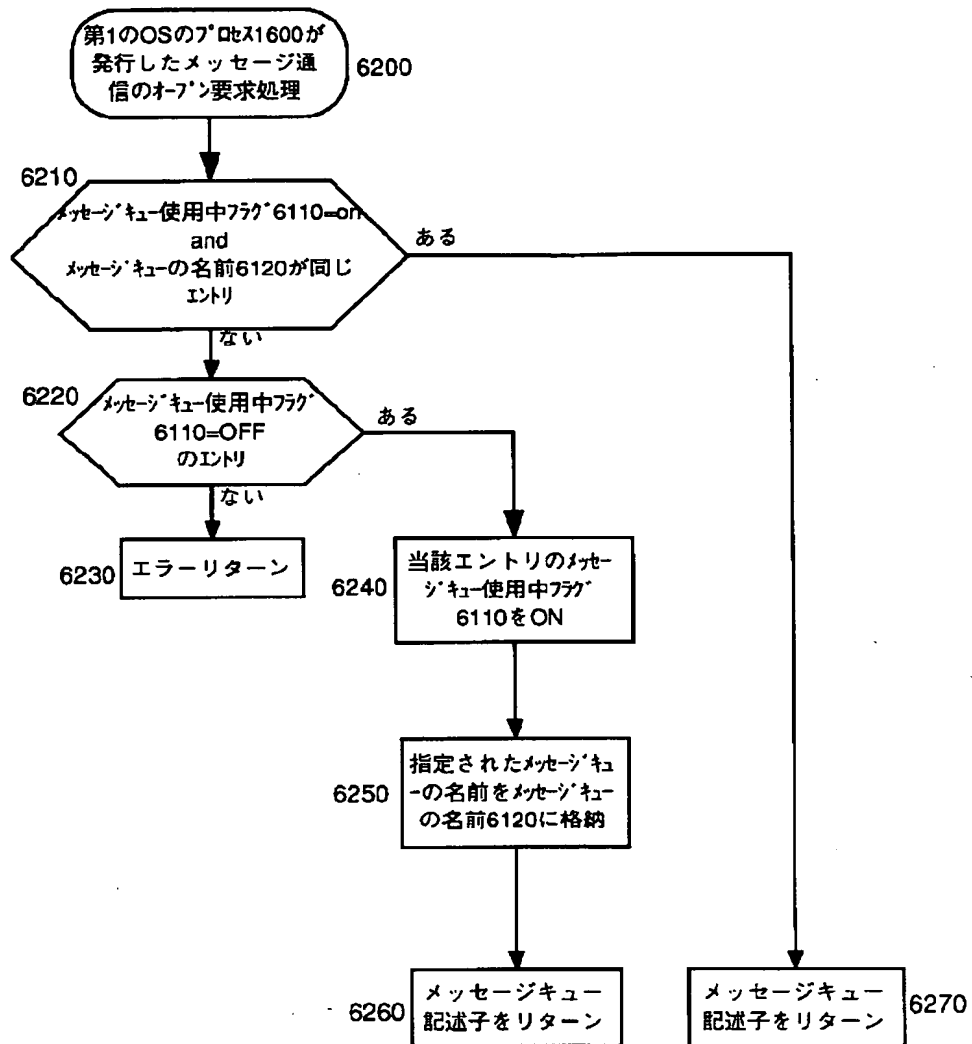
【図20】

図20



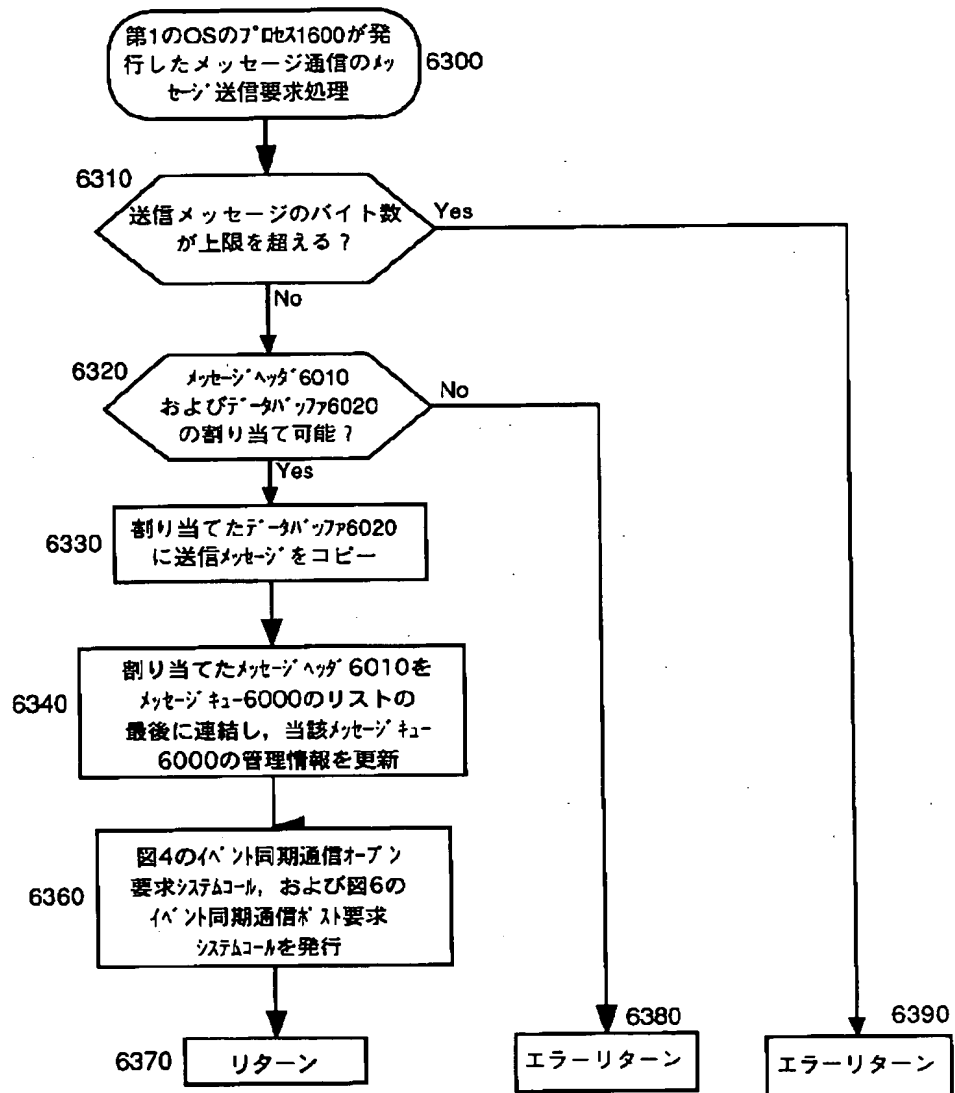
【図 23】

図 23



【図 2 4】

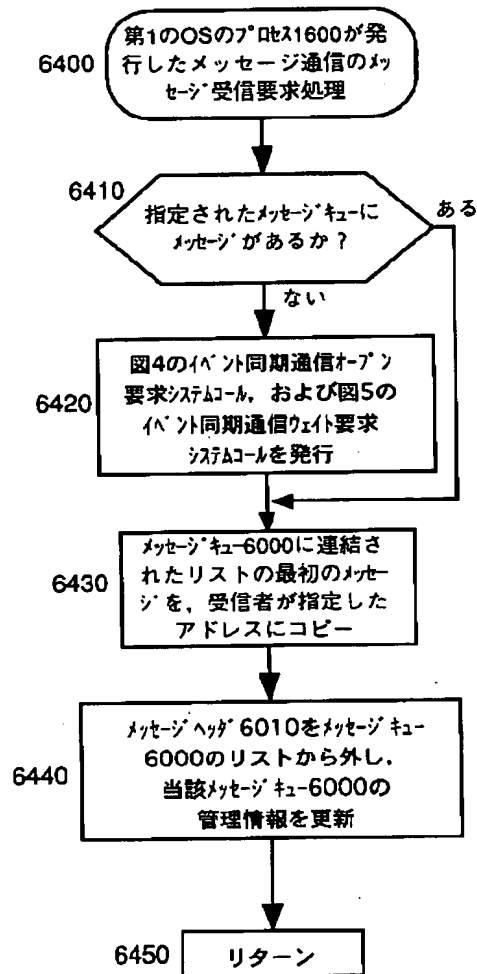
図 2 4





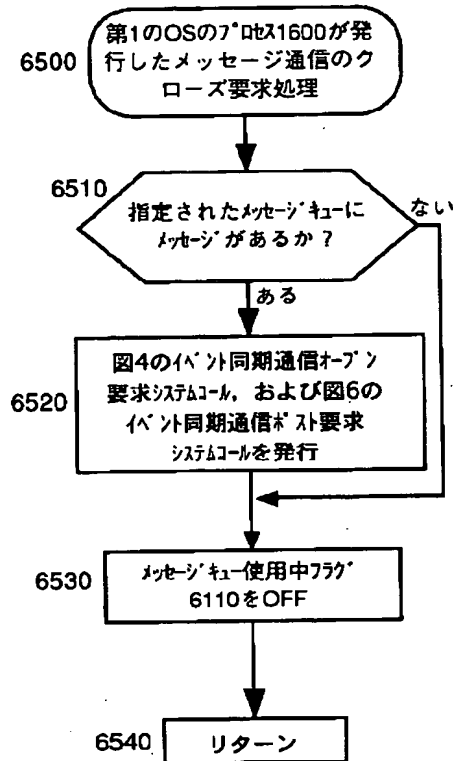
【図 25】

図 25



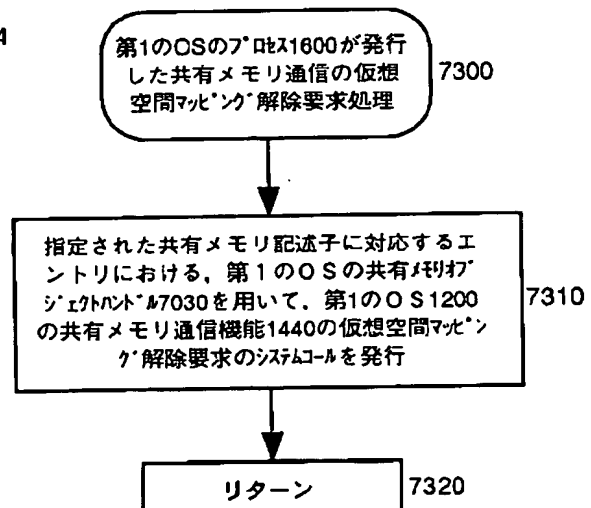
【図 26】

図 26

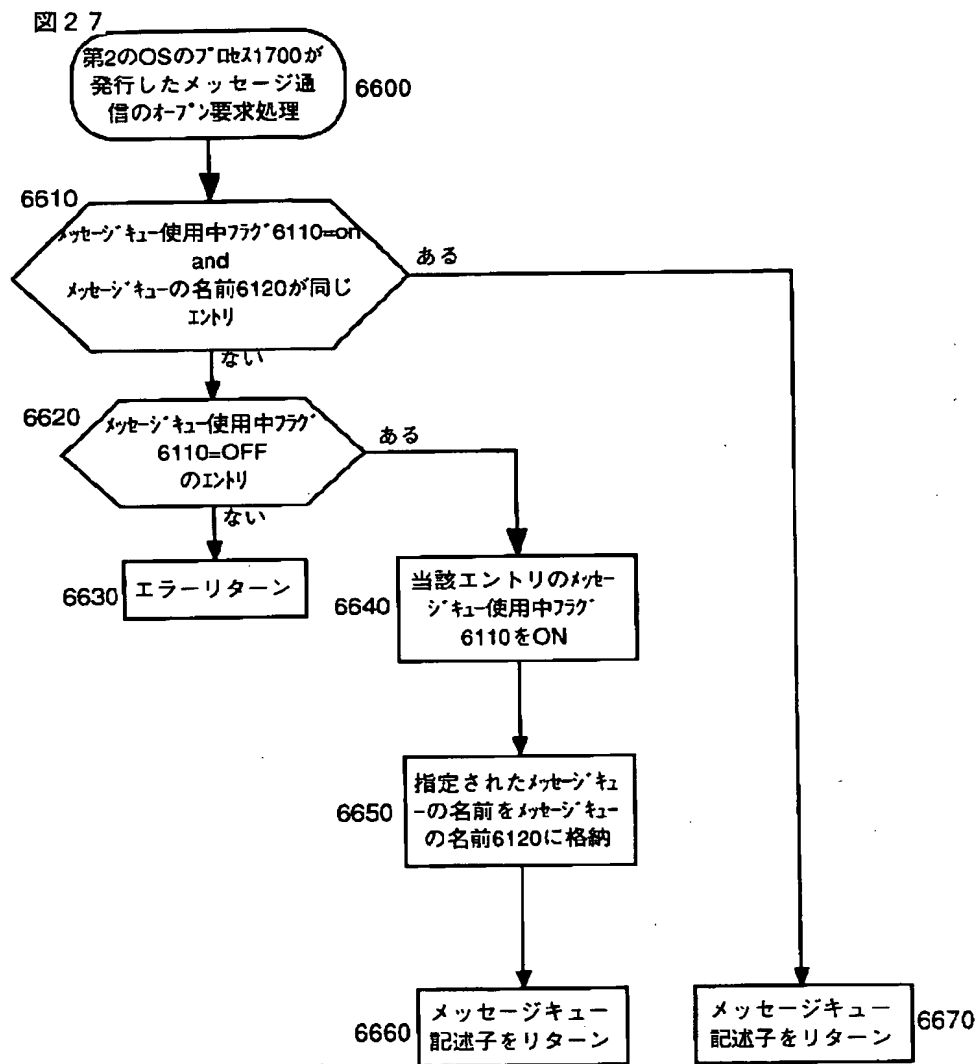


【図 34】

図 34

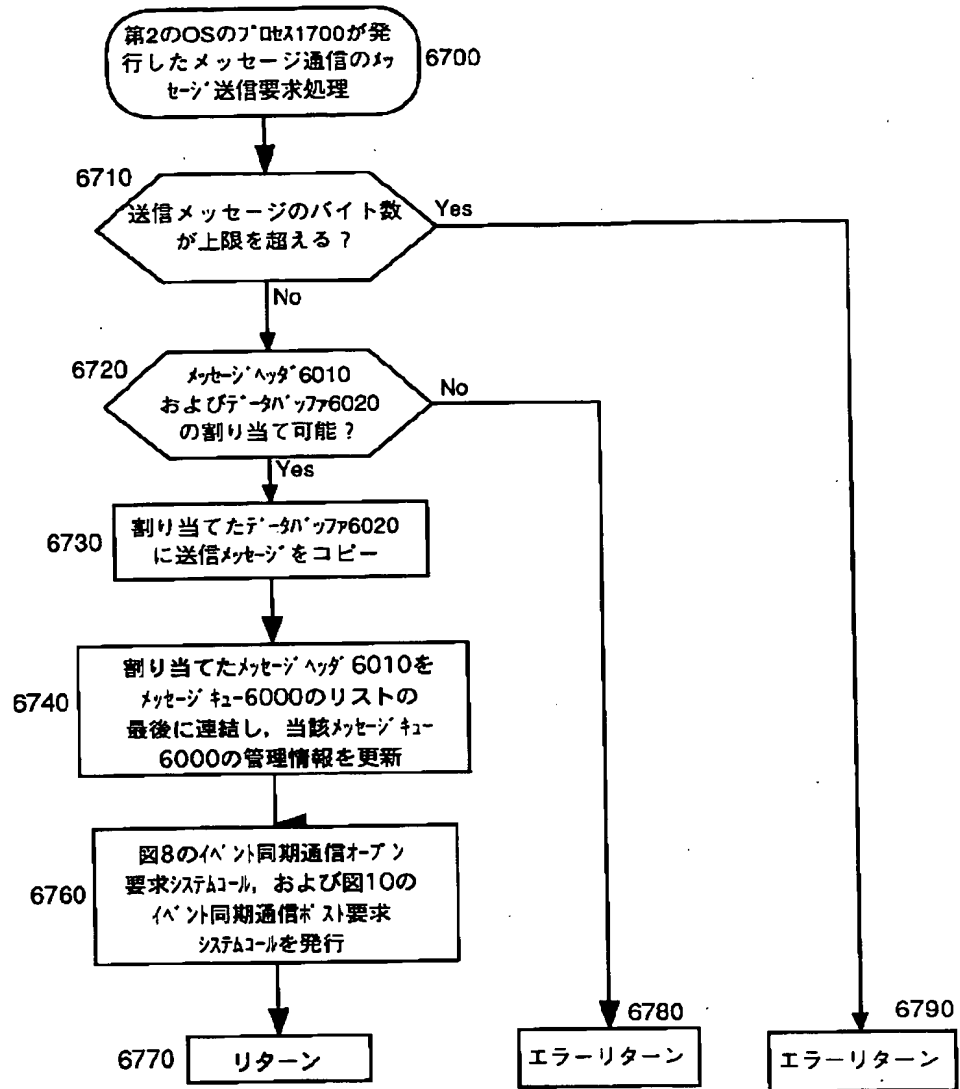


【図 27】



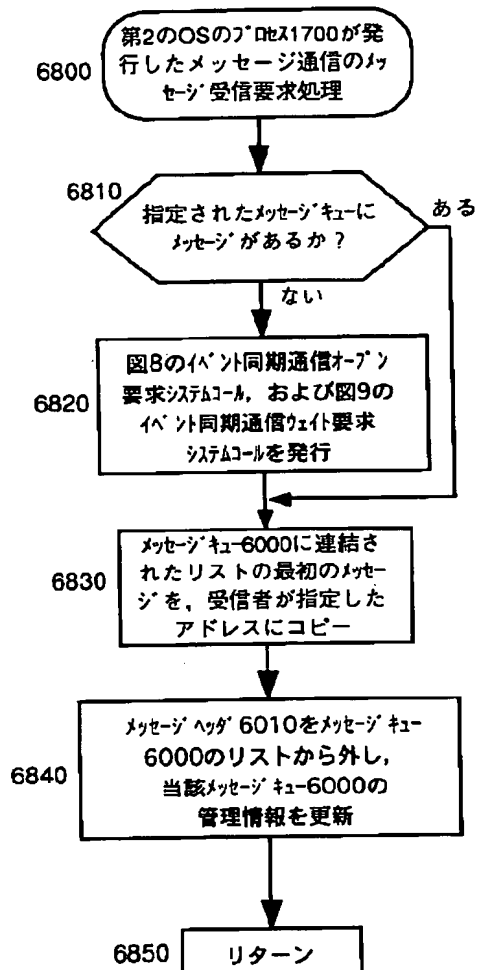
【図 28】

図 28



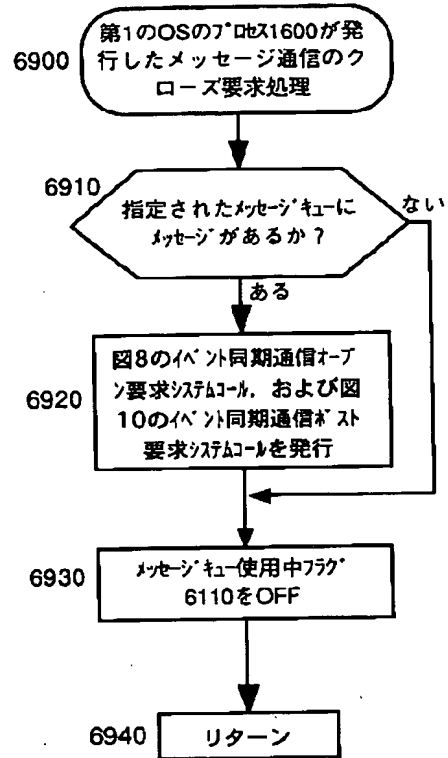
【図29】

図29



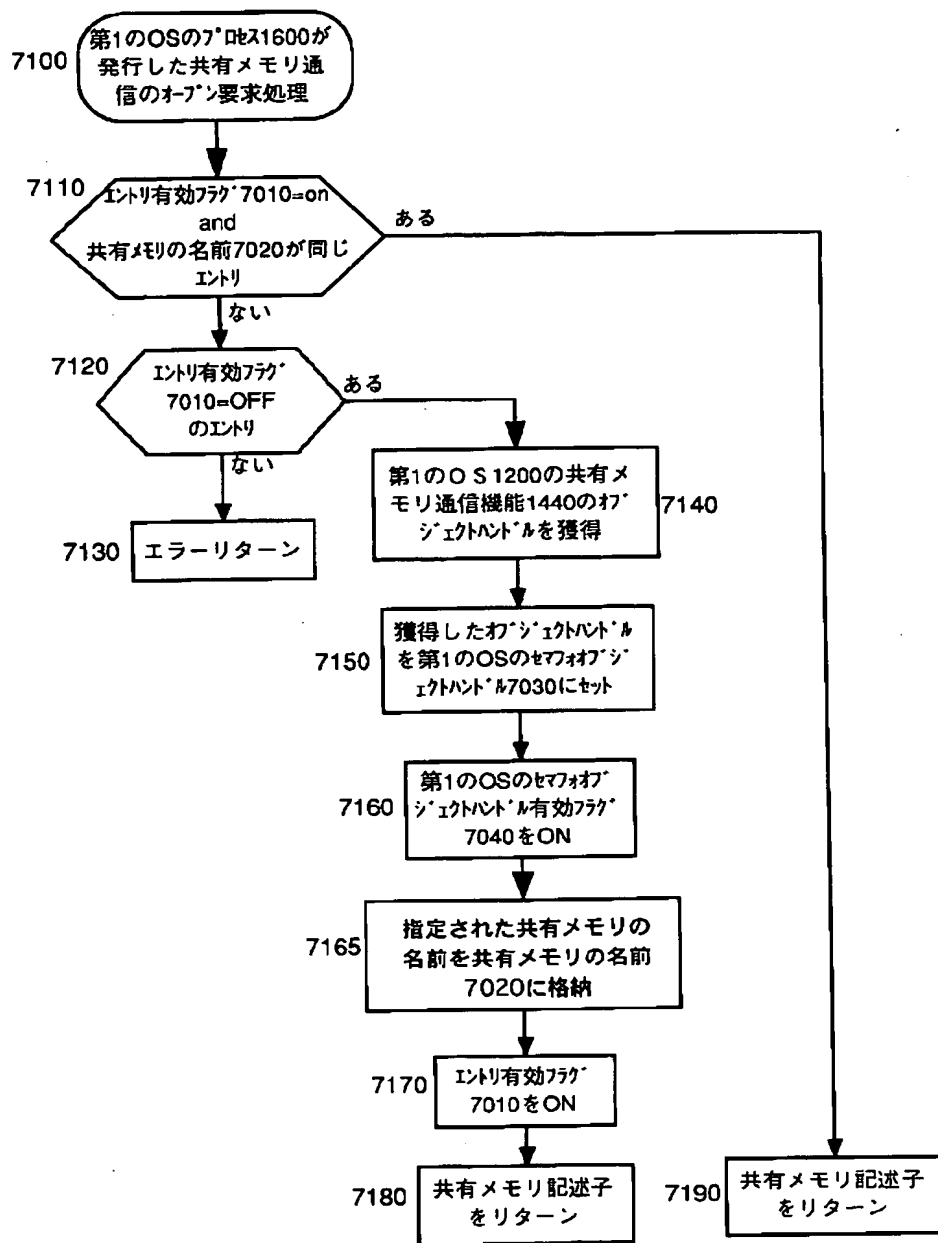
【図30】

図30



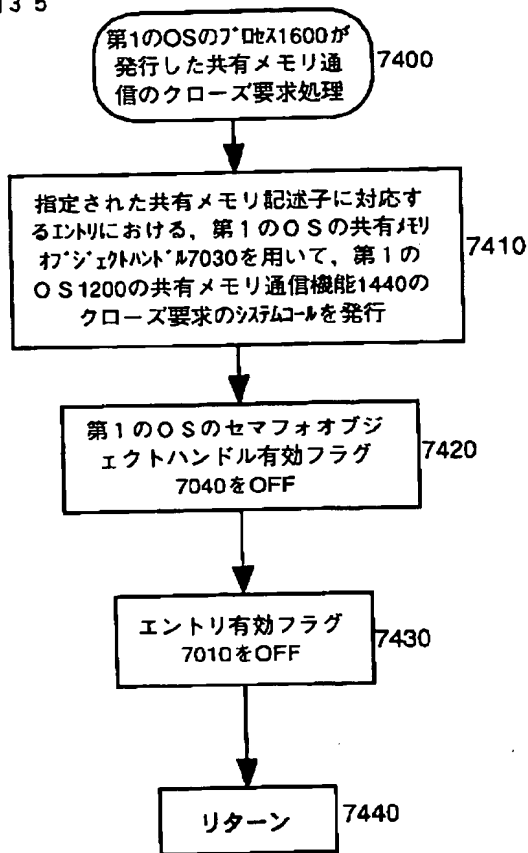
【図32】

図32



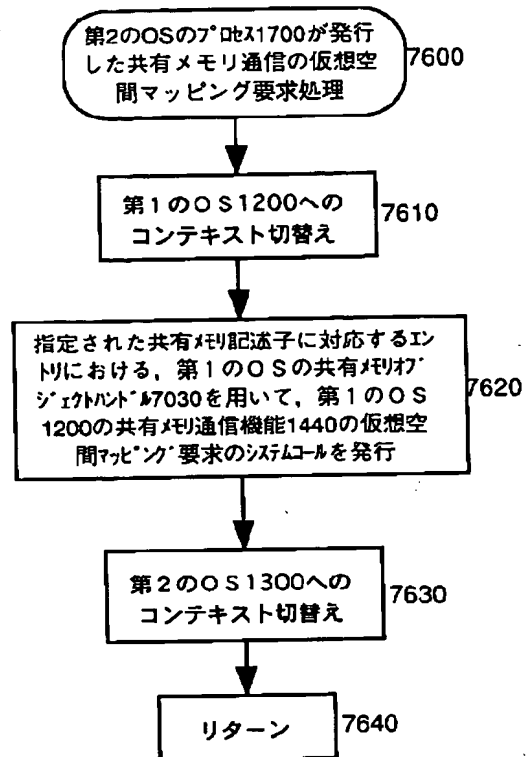
【図 3 5】

図 3 5



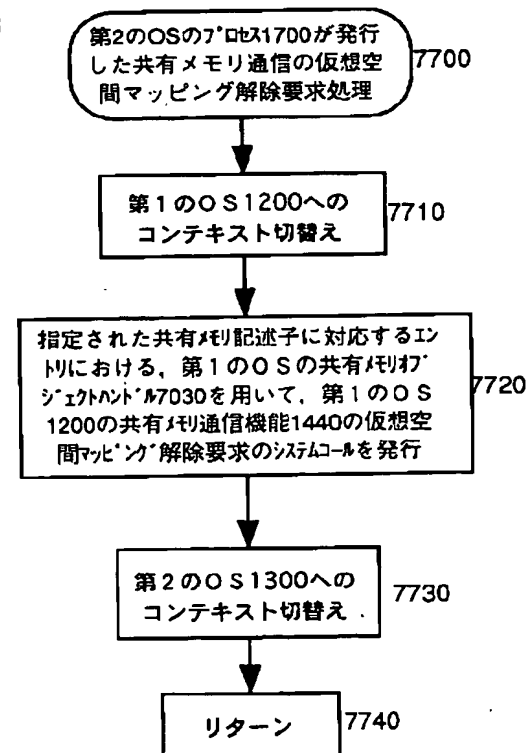
【図 3 7】

図 3 7



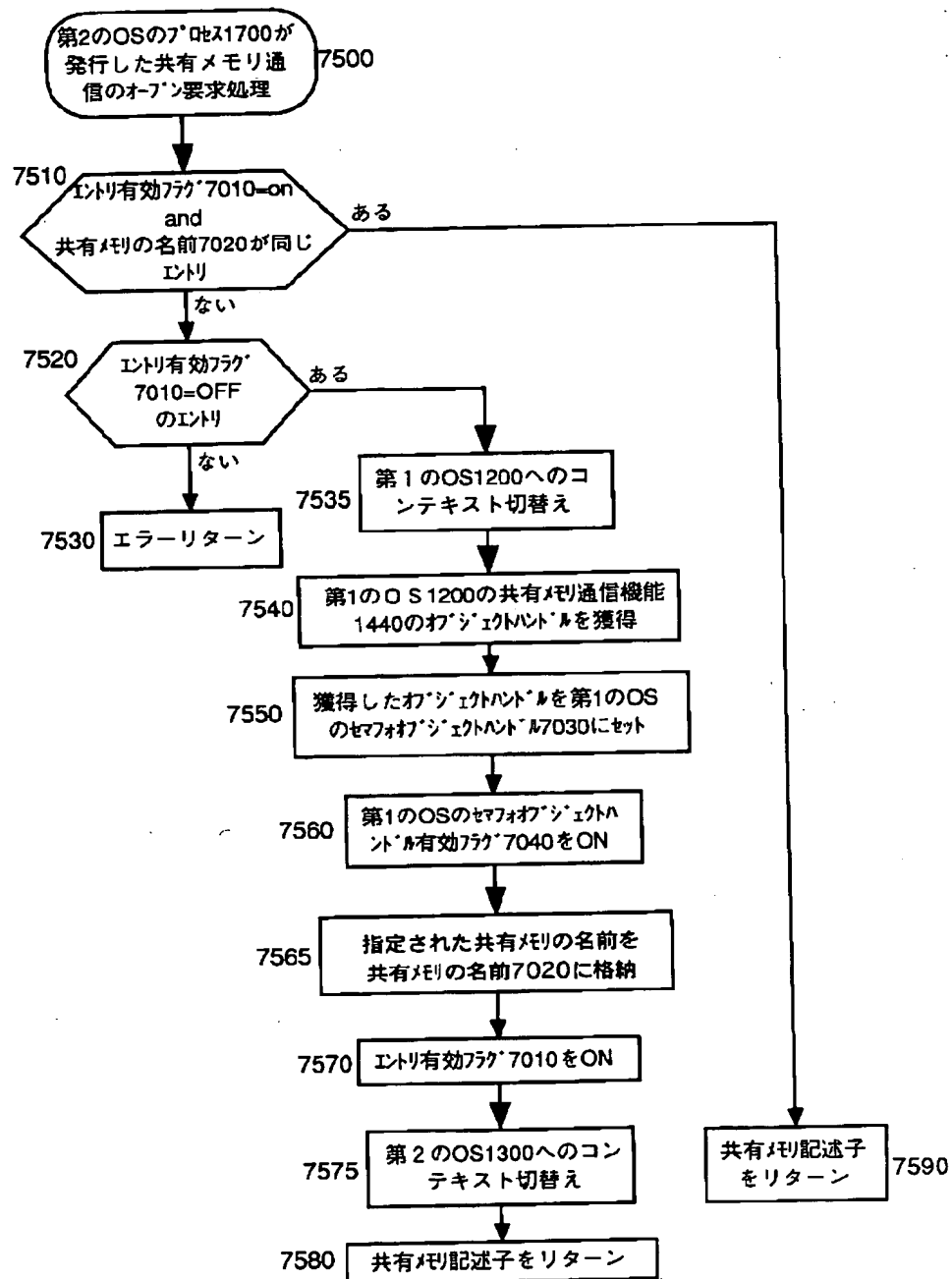
【図 3 8】

図 3 8



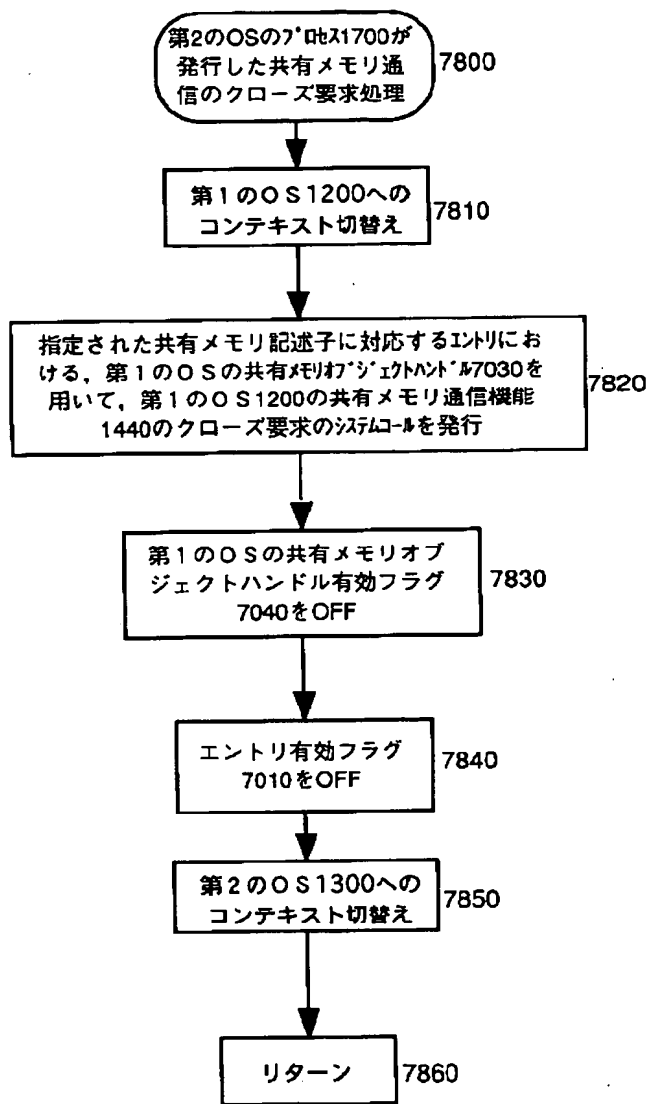
【図 3 6】

図 3 6



【図 39】

図 39



フロントページの続き

(72)発明者 大野 洋  
茨城県日立市大みか町五丁目2番1号 株  
式会社日立製作所大みか工場内

(72)発明者 関口 知紀  
神奈川県川崎市麻生区王禅寺1099番地 株  
式会社日立製作所システム開発研究所内  
(72)発明者 柴田 隆  
神奈川県横浜市戸塚区戸塚町5030番地 株  
式会社日立製作所ソフトウェア開発本部内